2012

Virtual Worlds and Conservational Channel Evolution and Pollutant Transport Systems (Concepts)

Chenchutta Denaye Jackson

VIRTUAL WORLDS AND **CON**SERVATIONAL **C**HANNEL **E**VOLUTION AND **P**OLLUTANT **T**RANSPORT **S**YSTEMS (CONCEPTS)

A Dissertation
Submitted to the Faculty of the University of Mississippi
in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy in the School of Engineering
The University of Mississippi

by

CHENCHUTTA DENAYE CROSS JACKSON

July 2012

ABSTRACT

Many models exist that predict channel morphology. Channel morphology is defined as the change in geometric parameters of a river. Channel morphology is affected by many factors. Some of these factors are caused either by man or by nature. To combat the adverse effects that man and nature may cause to a water system, scientists and engineers develop stream rehabilitation plans. Stream rehabilitation as defined by Shields et al., states that "restoration is the return from a degraded ecosystem back to a close approximation of its remaining natural potential" [Shields et al., 2003]. Engineers construct plans that will restore streams back to their natural state by using techniques such as field investigation, analytical models, or numerical models. Each of these techniques is applied to projects based on specified criteria, objectives, and the expertise of the individuals devising the plan. The utilization of analytical and numerical models can be difficult, for many reasons, one of which is the intuitiveness of the modeling process. Many numerical models exist in the field of hydraulic engineering, fluvial geomorphology, landscape architecture, and stream ecology that evaluate and formulate stream rehabilitation plans. This dissertation will explore, in the field of "Hydroscience", the creation of models that are not only accurate but also span the different disciplines. The goal of this dissertation is to transform a discrete numerical model (CONCEPTS) into a realistic 3D environment using open source game engines, while at the same time, conveying at least the equivalent information that was presented in the 1D numerical model.

DEDICATION

First, I would like to give all honor and glory to my Almighty God. He has made all my impossible possible. I dedicate this paper to my supportive family Marlon, Kentarvis, Christopher, and Aiden. You guys have made this journey worthwhile. Also I dedicate this to my grandmother Rosie Lee Cross. Without your faith and love for me, I would not be who I am today.

ACKNOWLEDGMENTS

I would like to take the time to express my deepest appreciation to my advisor, Dr. Pamela Lawhead and my committee members, Dr. Dawn Wilkins, Dr. Yixin Chen, Dr. Conrad Cunningham, and Dr. Eddy Langendoen. I would like to also express my gratitude to the University of Mississippi Graduate School for patience and funding as well as the United States Department of Agriculture for their commitment and involvement. I would like to thank my friend Doris Turnage for her support and advice throughout this entire process. Lastly, I would not have gotten this far without the support and help of my loving husband Marlon Jackson. You have stood by my side and loved me through it all.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE PAGE

CHAPTER I

INTRODUCTION

Channel morphology is defined as the change of the shape of an alluvial stream (river) over time. Stream restoration, rehabilitation, and reclamation projects are constructed to reverse the effects of adverse changes to streams. The two processes that contribute to the changes that occur in alluvial streams are water flow and sediment transport. Rapid variations in water flow and sediment discharge are caused by either man-made or natural disturbances to the ecosystem. Some of the man-made disturbances are deforestation, urbanization, irrigation, and vegetation development. The natural changes that arise in the ecosystem are precipitation increase, rainfall, and global warming.

Other common man-made disturbances that cause disruption in stream function that are often overlooked are chemical induced disturbances. Chemically defined disturbances can be introduced through many activities including agriculture (pesticides and nutrients), urban activities (municipal and industrial water contaminants), and mining (acid mine drainage and heavy metals)[The Federal Interagency Stream Restoration Working Group (FISRWG), 1998] [Wagner, Marsalek, and Ji, 2008]. Chemical disturbances do not just affect the area where they are released; their impact can be felt far downstream. Introducing exotic species to a particular area is also another common man-made disturbance to the streams ecosystem. The introduction

of exotic species, whether intentional or not, can cause disruptions such as predation, hybridization, and the introduction of diseases.

Agricultural overuse of the land is considered a man-made disturbance. For instance, overuse of the land by grazing cattle and sheep is universal throughout the country. Excessive grazing of livestock poses many problems such as landscape breakage, increased runoff, reduced infiltration, and physical damage to a stream corridors' vegetation.

Natural disturbances to streams are caused by tornados, hurricanes, flood, fire, lightning, volcanic eruptions, earthquakes, insects, diseases, landslides, temperature extremes, and drought. Studies have found that nature induced changes seldom need human intervention to restore a stream corridor back to its innate state [FISRWG, 1998]. In the case where humans have to intervene, stream restoration, rehabilitation and reclamation plans are devised.

Steam *restoration* as defined by the Federal Interagency Stream Restoration Working Group, is the reestablishment of the structure and function of an ecosystem [National Research Council, 1992]. Ecological restoration is the process of returning an ecosystem as closely as possible to pre-disturbance conditions and functions. Implicit in this definition is that ecosystems are naturally dynamic; it is therefore not possible to recreate them exactly. The restoration process reestablishes the general structure, function, and dynamics but permits self-sustaining behavior of the ecosystem.

The Federal Interagency Stream Restoration Working Group defines stream *rehabilitation* as making the land useful again after a disturbance [The Federal Interagency Stream Restoration Working Group, 1998]. It involves the recovery of ecosystem functions and processes in a degraded habitat. Stream rehabilitation and stream restoration are used

interchangeably; the main goal is always to make the land useful again after disturbances have been eliminated. Stream *rehabilitation* plans are the process and range of action taken to restore dynamic equilibrium and function to the stream corridor, which will allow it to be self-sustaining. Stream reclamation is a series of activities intended to change the biophysical capacity of an ecosystem. The main goal of these management methods is to restore the river or stream channel back to its self-sustaining ecosystem

Individuals from many fields such as ecology, biology, geomorphology, geology, landscaping design, hydrology, and hydraulics are all involved in the process of creating and implementing stream restoration, rehabilitation, or reclamation plans. The focus of this dissertation will be on stream restoration efforts as it pertains to hydrology and hydraulics. Hydrology is the study of water and all its complexity. Hydraulics is an area of applied science that is concerned with fluid properties and fluid flow. The engineers and researchers involved in developing stream restoration plans evaluate how water flows and how sediment transport operates on the stream channel. They are concerned with channel morphology and the geomorphology process. Many of the plans that hydraulic engineers devise have as an objective the stabilization of the banks and beds of a stream. This is achieved by providing the unstable bank with native vegetation along the riparian zone of the streams. Another method generally used is engineering the stream, either by adding riffles or pools or man-made hydraulic structures such as dams, rocks, cobbles, and culverts.

Based on research and engineering practices, regardless of the area of discipline, successful rehabilitation, restoration, and reclamation plans are derived based on three methods identified by Rouse [US Army Corp of Engineers, 1996]. These methods are viewing and

collecting past and present data from a stream using gauging stations, construction of physical models of a stream, and analysis using numerical models. Benefits and limitations exist for each of the methods.

The oldest technique used for planning and developing stream rehabilitation projects requires engineering experience. Field experience is highly important and is an extremely valuable tool for any scientist or engineer. Collecting field data is one of the steps in the process of devising a stream restoration plan. For stream restoration projects, individuals involved must access, analyze, and compare gauging station data and historical data that have been stored in repositories on the steam and stream corridor under investigation. The major drawback of this technique is that it does not always provide defensible and reproducible results. Another disadvantage is that this method often leads to trial-and-error procedures. Collecting data from gauging stations and comparing it with historical data is considered to be a labor intensive process. Also the information from these gauging stations is considered to be harder to obtain and predict accurately.

The second technique is based on physical models. These models are constructed to replicate rivers or channels being analyzed. Laboratory modeling has been proven to be successful and dependable for at least the past 60 years [US Army Corp of Engineers, 1996]. The advantage of physical modeling compared to the first technique mentioned is that it is mandatory that all physical model procedures be documented. The major drawback of using the laboratory or physical modeling technique is that it can become very costly. Because of the many variables that exist in open channels, physical models are not considered feasible by many river engineers and scientists who practice the first method. Based on their assessment, many plans devised for

alluvial channels using physical models for analysis fail, because they are often too costly or their results are ineffective.

The last technique in planning stream rehabilitation projects are analytical procedures based on mathematical procedures and numerical models [US Army Corp of Engineers, 1996]. Research shows that the most efficient, cost-effective, and stable stream rehabilitation plans were devised using results from numerical modeling. Numerical models can be classified as one, two, or three-dimensional, and as hybrid models. Users of these models may choose one or a combination of models that will yield successful and defensible results at an optimal cost.

Models that are considered when devising stream or river restoration projects are fluid-flow models and channel evolution models. There are many advantages to using these numerical models. The mechanics of flow and sediment transport have been studied and investigated in many field studies, flume experiments, and theoretical studies. Numerical channel evolution models allow researchers to retrieve results of proposed plans quickly. Evaluating project alternatives is very cost-effective. These computer models can provide scientists and researchers with accurate and quick results on how a river's morphology will change over time. Not only do the models predict the future equilibrium morphology of a channel under evaluation, they also show the temporal evolution of the channel morphology. These models allow scientists, engineers, and researchers to understand the long-term, broad scale evolution of channels and floodplains [Howard, 2008].

In hydraulic engineering, numerical models have the capability of simulating fluid flow processes and channel evolution efficiently and accurately. Often fluid-flow models, channel evolution and sediment transport models are used in combination to form a more powerful

computational tool. The results from these multiple models provide greater insight into the dynamic channel evolution and fluid-flow processes. These tools provide stream restoration developers with information on how fluid flow affects the stream's geometry (e.g., width/depth ratio). The primary limitation of these tools is that the results from these models may be difficult to understand and results are hard to decipher for both expert and novice users.

Tools created to enhance these scientific models are created with visualization capabilities. Research shows visual stimuli provides users with a better perception of the simulated scenarios given by computational models. Scientific and information visualization have been used for many years in all areas [Schroeder, Martin, and Lorensen, 2004] including education, military, industrial, and scientific research. Visualization based on computer graphics and interactive simulation techniques was formally defined 20 years ago [Rhyne, 2007].

Solutions that exist to enhance numerical models used for stream restoration management involve various analytical and visualization technologies. These numerical tools used for developing stream restoration plans are combined with GIS analysis tools, powerful Graphic Processing Units (GPUs), and innovative gaming engines. Each technique used to enhance the numerical model has benefits as well as limitations.

The most popular techniques among the scientists who devise stream restoration plans are the methodologies that combine GIS analysis technology. GIS analysis tools and techniques span many areas of environmental and agricultural science. Due to the popularity of these tools there are many readily available.

GIS tools that are combined with numerical models for environmental management are found as both proprietary and open source software. Researchers in the field of environmental

science have an easy task of using the tools because many tools that exist are domain specific for this area of science. Individuals who have neither background knowledge nor experience with numerical models nor GIS tools find them difficult to navigate, let alone understand, the valuable information provided from these tools. Because of this, the combination of these many tools results in an even greater level of difficulty of use.

Another technique that is used to enhance the numerical models used for stream restoration is using powerful supercomputers and GPU technology. Increasing CPU power and using highly effective graphics hardware have always been found to enhance many tools used for research purposes. The major drawback with this technique is that funding for projects of this caliber are not always feasible and are always costly. Technology, at this level, is expensive and requires experts in the field of each technology to design a tool that is effective and domain specific but also user friendly.

The latest technique used to enhance numerical models used for stream restoration management is the addition of gaming engines to visualize the data from channel evolution and fluid-flow models. Gaming engines are complex software systems designed for creating and developing video games. The main function of a gaming engine is to efficiently render 2D and 3D graphics. According to Lewis and Jacobson, the only way to have the fastest, most realistic simulation and sophisticated graphics is to trade down from the expensive hardware to basic PCs running game software [Lewis and Jacobson, 2002]. The technique of combining the two restoration management tools with a gaming engine combats the issues presented by the first two techniques. Although this area of science is new it offers great potential with advancing research efforts in the area of stream restoration [Guo et. al, 2008] [Wells, 2005] [He et al., 2006].

There are many benefits to be derived from using gaming engines. Most of these options are fairly inexpensive. There are primary categories for a gaming engine. Both proprietary and open source game engines exist. Choosing one over the other makes little difference but increases the potential of developing a great tool within many budget sets for stream restoration planning efforts. So, the costs associated with developing tools with the mechanisms that gaming engines provide are minimal. The features geared toward visualizing large datasets such as terrain data, already exist and are readily available. Also many projects developed with gaming engines have been shown to be easy to navigate and the results from these models are easy to understand [Lewis, and Jacobson 27-31]. The major drawback with this technology is that, while popular in many other areas of science, it is rarely used in the area of environmental planning for stream restoration programs. This limits the documentation available for creating programs that are domain specific. Also the implications of developing these tools are unknown. The largest drawback known when using this technology is that visualizing or rendering very large terrains can cause time and space issues. But as CPU power and storage are increasing this will become less of an issue. The possibility of creating tools that are cost effective, domain specific, and user friendly will only make the use of gaming engine technology within this area of study more promising.

This last method, numerical data set rendering using a gaming engine, was the technique of choice for this dissertation. This method was chosen because the costs associated with producing a tool to enhance the numerical model used for stream restoration planning were free. Also, the efficient productivity of creating software with a gaming engine was considered to be effective due to the availability of many, open source tools. Many features for depicting data

graphically are already a built-in function of many gaming engine technologies. Basically, time and money were the driving factors for considering using open source gaming engines to develop a tool that was user friendly and effective for displaying the data from the existing numerical model (CONCEPTS) making it a visual tool that is intuitive, allowing many users to navigate. A secondary goal was that visualizing the large data sets resulting from numerical modeling will be more understandable thus making the models more effective.

The objective of this dissertation is to enhance the one-dimensional, numerical model, Conservational Channel Evolution and Pollutant Transport System (CONCEPTS) model with graphical capabilities provided by a gaming engine [Langendoen, 2000]. The ultimate goal is to transform the discrete numerical model into a realistic 3D environment, while at the same time conveying the equivalent, accurate information that was presented in the 1D model.

CONCEPTS is a computer model used to simulate open-channel flow, sediment transport, and channel morphology. The primary aim here is to develop an additional component/dimension for CONCEPTS that provides users with an intuitive visual, approach to evaluate stream restoration projects graphically. Currently, CONCEPTS does not produce an intuitive output, its input and output is in the form of text which is imported into an EXCEL spreadsheet (see Figure 1).

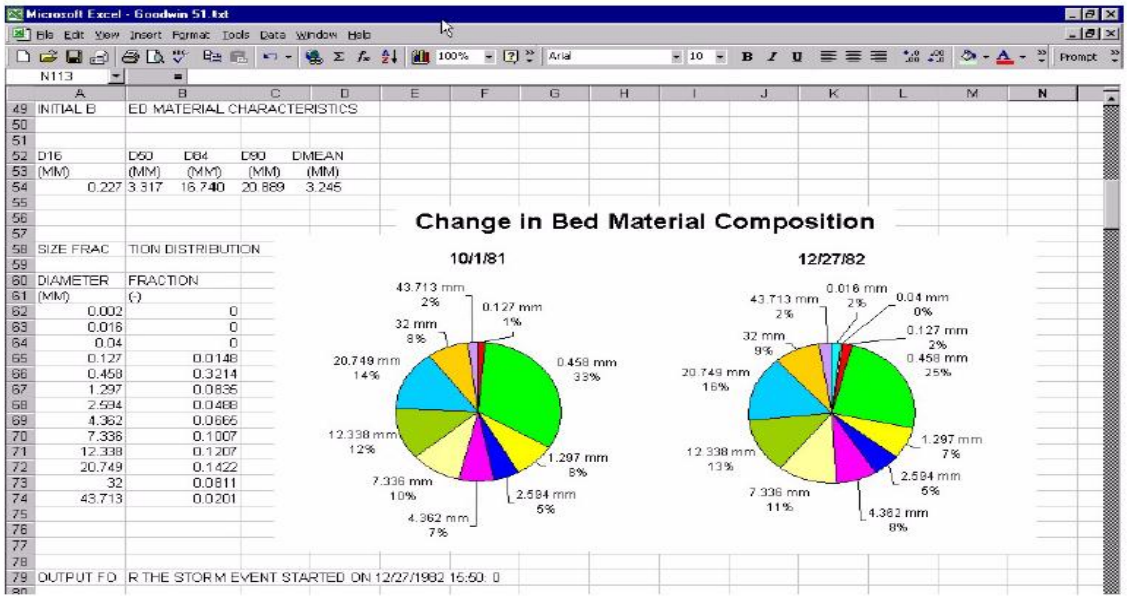


Figure 1: Source NSL-Technical Report 16 [Langendoen, 2000]

Providing a prototype for real-time, graphical interface feature to CONCEPTS will open the door for many more users besides hydro-scientists to understand and use the modeling tool. The results from the new model CONCEPTS3D will be easier to understand for both novices and experts. In order to accomplish this goal other objectives of this dissertation must be met. These include:

1. Build a desktop application available for non-sophisticated users, which visualizes CONCEPTS data in 3D-CONCEPTS3D.

2. Store and access CONCEPTS XML data through a MySQL database.

3. Develop three parsers to extract appropriate data to model a 3D virtual environment.

    a. Parser—XML file
    b. Parser—DEM file-terrain
    c. Parser—Cross-section (X-Section) file-channel

4. Transform DEM and X-Section data into a height-map file format.

5. Develop data interpolation techniques that allow for CONCEPTS data to be available for the 3D gaming engine.

6. Convert interpolated data into gaming engine native file format.

7. Find and adapt an opened source gaming engine so that it can be repurposed for CONCEPTS viewing.

Each component created to achieve the listed objectives will be discussed and reviewed in greater detail in the methodology section of this paper.

CHAPTER II

LITERTURE REVIEW

In this Chapter, an assessment of 1D numerical models and enhanced models with visualization capabilities is presented. Each of the first set of models discussed is used to analyze sediment transport, channel morphology, and fluid flow rate. The numerical models reviewed are Generalized Sediment Transport Model (GSTARS), Hydraulic Engineering Center (HEC-6/HEC-RAS), National Center of Computational Hydroscience and Engineering (CCHE1D), and Conservational Channel Evolution and Pollutant Transport System (CONCEPTS). These models are limited by the lack of visualization and graphics capabilities they provide. The one model enhanced with visualization efforts for stream restoration planning is CCHE1D. This model visualizes its data using an ArcView program. Models that incorporate Graphic Processing Units (GPUs), GIS, and gaming engines technology to enhance the graphical and visualization capabilities of applications and models are assessed.

GSTARS

GSTARS was developed by the United States Bureau of Reclamation [Yang et al.]. GSTARS is used by government agencies, researchers, institutions, students as well as engineers for the analysis of sediment transport. The model is used for analyzing sediment, erosion, morphology, and river restoration [Yang et al.].

GSTARS 1-D is a model presented under series GSTARS models. GSTARS 1-D is a one-dimensional hydraulic and sediment transport model for use in natural rivers and man-made canals. This numerical model is used for steady, unsteady flow analysis, internal boundary conditions, looped river networks, cohesive and non-cohesive sediment transport, and lateral inflows analysis.

GSTARS 1-D was applied by Klump (2005) to simulate unsteady sediment transport in the California aqueduct [Yang et. al]. The results were significant. It was found that the model provided information that the channel had aggraded more than 136 miles downstream.

GSTARS is a general numerical model and just as other 1-D models, it has its limitations. Limitations include:

1. The model requires expert evaluation of results to certify that the model's predictions are correct.
2. Currently the system GSTARS can be executed only on a Windows 2000/XP.
3. The size of the project evaluated can be limited only to the amount of computer memory space available.
4. It is stated in the manual that the program results are potentially fallible;
5. The results should be examined by an experienced engineer to determine if they are reasonable or accurate.

GSTARS software is available online for download at http://www.usbr.gov/pmts/sediment/model/srh1d/index.html. The current model reads input files, which are organized as sequential records. To execute GSTARS1-D, the user is required to enter the file name with the extension .srh. During execution of the program the current bed profile and user specified cross-section is displayed in real-time. During the simulation process, GSTARS' output allows the user to visualize the results of the input data.

Currently, GSTARS is known as Sedimentation and River Hydraulics-One Dimension

(SRH-1D).

HEC-6

HEC-6 is a software package developed by the U.S. Army Corps of Engineers (USACE) Hydrologic Engineering Center (HEC) [US Army Corps of Engineers, 2009]. The purpose of this one-dimensional sediment transport model is to calculate water surfaces and sediment bed surface profiles by computing the interaction between sediment material in the streambed and the flowing water-sediment mixture. HEC-6 has the capability of analyzing a network of streams, channel dredges, and various levee encroachment plans. This model also provides several methods to compute sediment transport rates.

HEC-6 has been applied by many organizations besides the USACE offices. HEC-6 has been used for government, private consultant, university, and foreign projects. One project the model was applied to was to predict potential future sedimentation for the Ozark Reservoir on the Arkansas River. HEC-6 was also used to predict future water surface elevations for levees at Lewiston ID on the Lower Granite Reservoir. HEC-6 was applied to other projects described by Thomas and Prasuhn (1977), and other recent projects of HEC (1992). The HEC-6 model has been used extensively since the 1970's for predicting sediment transport rates, dredges and levee plans accuracy, as well as channel morphology.

Although HEC-6 has been used for various reasons, the model does have its limitation. HEC-6 model is restricted by the following limitations:

1. The model has no provisions for simulating the development of meanders.
2. The model does not provide provision for simulating lateral distribution of sediment loads across a cross-section.

3. Density and secondary current are not simulated.

4. Movable beds are constrained within the limits of wetted perimeters. In the case of sediment transport HEC-6 is restricted in two ways.
    1. First sediment transport in distributaries is impossible.
    2. Flow in closed loop channels can not be directly accommodated.

5. Only local inflow points are allowed between any two cross-sections for calculating sediment transport (US Army Corp of Engineers).

HEC-6 is also available online for download at

www.hec.usace.army.mil/software.legacysoftware/hec6/hec6.htm.

CCHE1D

CCHE1D is a model developed at the National Center of Computational Hydro Science and Engineering at the University of Mississippi [Vieira and Wu, 2002]. The CCHE1D modeling system was created to simulate steady and unsteady flow and sedimentation processes in dendritic channel networks. The model has the capability to simulate sediment transport, bed aggradation and degradation, bank erosion, and channel morphology. One limitation, however, is that the model is restricted to the application of sub-critical flows [Vieira, 2002].

The CCHE1D model has been applied to several waterway projects [Vieira, 2002]. CCHE1D was applied to East Fork River, Wyoming, a sedimentation project in the Danjiangkou Reservoir, Goodwin Creek watershed in Mississippi, and the PA-Chang river flood routing and sediment transport project in Taiwan.

The features provided by CCHE1D are innovative for 1D models. CCHE1D has an ArcView graphical user interface. This feature allows the user of the model to follow a natural flow of operation. The CCHE1D ArcView interface also gives users more options for defining the simulation domain and input data. The model includes a "Channel Digitizing Module,"

which allows the user to sketch channel networks based on maps or photographs. Other features are a Landscape Analysis and Channel Network Analysis module. The Landscape Analysis module creates a channel network based on digital elevation data. An upgraded version of Topographic ParameteriZation (TOPAZ) is integrated in CCHE1D. TOPAZ analyzes a Digital Elevation Model (DEM) and extracts the channel networks and corresponding subcatchments.

The use of the CHE1D model requires knowledge in the fields of hydrodynamics and channel morphology. First, the user must select the StartChannelSimulation from the Simulation menu. The model will then analyze the input file. After this step the progress of the simulation is displayed. When the simulation has completed, a message indicating the end of the process is displayed. The results of a CCHE1D simulation are easily read and displayed, either in a visualization, a spreadsheet, or data analysis program. One of the limitations of this numerical model is that it cannot be applied to dam break flow and it is not applicable to dendritic channels with more than one outlet. The visual of this program also uses GIS based graphic which limits it to 2D flyby image of the area.

CONCEPTS

CONCEPTS was developed at the United States Department of Agriculture (USDA)-Agricultural Research Service (ARS)-National Sedimentation Laboratory (NSL) in Oxford, Mississippi. CONCEPTS is a model in the suite of tools incorporated under Agricultural Non-Point Source Pollutant Model (AGNPS) [Langendoen, 2000].

CONCEPTS was designed as a 1-D numerical model used to simulate open-channel hydraulics, bank and bed erosion, and sediment transport. A primary objective of the CONCEPTS model is that it is to be used as an assessment evaluation model for proposed stream

or corridor rehabilitation plans.

The advantages of using CONCEPTS are that it can be applied to evaluate the effectiveness of instream control structures and stream corridor rehabilitation measures in controlling channel erosion. CONCEPTS has been used to study channel evolution in the Goodwin Creek, watershed, Mississippi and Little Salt creek of Eastern Nebraska, and TMDL studies in James Creek, Mississippi and Shades Creek, Alabama.

Although CONCEPTS provides scientists and engineers in the hydraulic field the necessary capabilities to evaluate and develop stream rehabilitation plans, the model has similar limitations as the models above.

CONCEPTS is freely available and can be downloaded at ftp://solar1.mas-oxford.ars.usda.gov/pub/outgoing/CONCEPTS. The requirement needed to execute CONCEPTS is a computer with Windows XP or later operating system. The CONCEPTS input file requires 1 to 2 MB of hard disk space. A minimum of 32 MB of Random Access Memory is recommended. The results of the model or output vary between 1MB to a couple of gigabytes depending on the output options provided by the user.

To execute CONCEPTS, the user is required to provide two types of input files.

1. The description of the channel, run control data, and output options are organized into a single XML input file.
2. Water and sediment discharge rates at the upstream boundary and from tributaries are organized into flat ASCII input files.

CONCEPTS produces three types of output files:

1. Output at a certain location and for a certain runoff event

2. Time-series output at a certain location

3. Output for a specified runoff event along a section of the evaluated and modeling reach.

Output from CONCEPTS must be imported into a spreadsheet application such as Microsoft Excel or Corel Quattro Pro to view the results of the simulation with plots or graphs (See Figure 2).

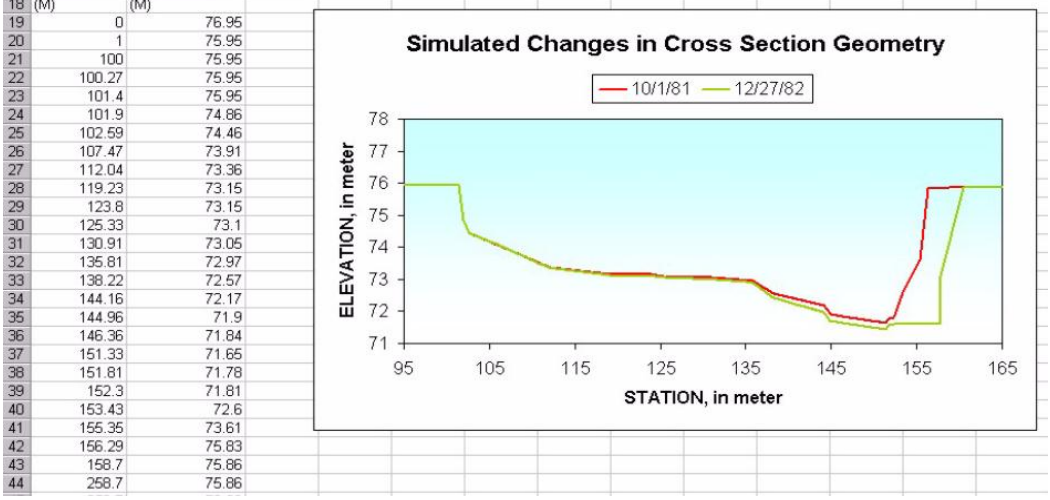


Figure 2: CONCEPT Output in EXCEL [Langendoen, 2000]

To overcome many of the limitations of numerical models, developers, researchers, and scientists have created many tools designed to be used with them. These tools utilize different technology that has built-in graphics and visualization features to increase usability and interpretation of results. Technologies that have been used to produce these tools are Geographic Information Systems (GIS), Graphic Processing Units (GPU), and gaming engines.

GIS based tools are the most popular enhancement tools used with numerical and mathematical models. GIS technology providing visualization capabilities as well as data

management and analysis has been improved for more than 30 years [Berry, 2007]. There are many tools that use this technology in many areas of sciences such as disaster and risk analysis and management, water quality monitoring and management, and forestry informization. Other areas that have also benefited from GIS technology are urban governance, simulation and environment modeling, pollution monitoring, traffic and transportation management, environmental and ecological marine monitoring and management, and analysis and simulation of farmland information systems. A few tools that provide GIS technology to existing applications and management systems are GIS for Transportation (GIS-T) [Lu, 2006], marine GIS (Chybicki et. al, 2008], 3D GIS and geo-analysis model [Guo et. al, 2008], and GIS-FIMS [He et.al, 2006].

GIS-T with web services was created by Xiaolin Lu at the College of Information Technology in Zhejiang, China [Lu, 2006]. GIS-T with web services is a distributed, platform independent system that can be accessed remotely over the Internet. The application provides efficient transportation planning and management formulas. GIS-T uses the latest in GIS technology web services in an effort to allow developers to create customized software. Using GIS web services gives developers the advantage of using traditional GIS application features without needing many modules or the standalone GIS applications and associated geographic data. The GIS web service of the GIS-intelligent transportation application system provides spatial data and GIS functionality to integrate the customized ITS applications to perform basic geo-processing tasks such as address matching, map image display, and routing [Lu, 2006].

The research performed by Guo, Zang, Zhao, and Ge at the Key Lab of Virtual Geographic Environment showed that integrating 3D GIS and geo-analysis was convenient and

efficient, and promotes geo-simulation [Guo et. al, 2008]. By using an existing 3D numerical model and incorporating 3D visualization of the geospatial (GIS) data permits for the contaminants transportation to be displayed and simulated. The application created was designed to simulate and visualize the three dimensions water contaminants transportation of the Taihu Lake [Guo et al. 2008].

Another tool that incorporates GIS technology with pre-existing software is Geographic Information Systems-Farmland Information Systems (GIS-FIMS) [He et al., 2006]. Created by He, Deng, Shao, and Fang at Zhejang University, GIS-FIMS is a tool used to monitor and manage farmland. GIS-FIMS uses short message technology to obtain the farmland information. The information acquired by the tool is the farmland electronic conductivity, temperature, moisture content, global positioning system (GPS) information, and PH levels. In essence the system uses GIS technology by integrating a short messaging service with web accessible capabilities that provides visualization of geospatial data, maps, and data services to the end-user.

The last tool discussed that incorporates GIS technology is marine GIS, and it is used in marine pollutant studies [Chybicki et. al., 2008]. This tool was created by Chybicki, Kulawiak, Lubniewski, Dabrowsi, Luba, Moszynski, and Stepnowski at Gdarisk University of Technology in Gdarisk, Poland. The tool provides remote access to a real-time management tool for processing and visualizing data from many sources such as sonar sensors, acoustic sensors and echo sounders. It supports instantaneous 2D and 3D visualization. This tool also uses many of GIS technology features such as ARCSDE, ARCGIS engine, GlobalControl, and map control components for geo-referenced presentation of objects. It also does the traditional features of

geo-processing and spatial analyzing. In essence the system was designed using many GIS technology features, which permits data to be processed and visualized from different sensors and distributed sources. The tool provided is a real-time, remotely accessible marine GIS application used for monitoring water pollution compounds and predicting the pollutants behavior in the environment.

With many of the GIS analysis tools created, providing them with visualization capabilities has been the key component that has been shown to increase not only the usability of the tools and computational models but also increased the clarity of information produced from computational models using the GIS features.

Research shows that use of visualization features has improved the decision-making process by showing analysis information that will save users effort, time, and money [Henderson & Mason, 2010]. The visualization and graphic component of many numerical models is the key feature that increases the usability as well as promotes clarity of the data being interpreted. Many engineers, researchers, and scientists are turning to incorporating, using, and studying GPUs (graphics processing units) and advanced graphics cards to amplify the visualization and imagery efforts of many analysis tools.

Although models that use GPU technology and advanced graphics cards increase usability, they are considered to be very computationally intense. Applications or tools created that use visualization in conjunction with numerical/simulation models require computationally intense programming that must be both stored and computed. To alleviate these constraints programmers, scientists, and researchers are looking at the latest technology that offers hope in the computer science and graphics industry. This method is co-processing of the central

processing units (CPUs) and graphic processing units (GPUs). This innovative technique was created to alleviate or offload some of the intense computation calculation to the graphics card installed in a user machine. Software engineers are now using Compute Unified Device Architecture (CUDA) technology to program scientific research applications on a GPU [“What is CUDA”].

CUDA is a parallel computing architecture developed by NVIDIA [“CUDA in Action”]. CUDA is the computing engine in the GPU presented by NVIDIA. This engine provides software developers access to the underlying hardware of a machine’s graphic card. CUDA gives developers access to the native instruction set and memory of the parallel computation elements in the GPU [“CUDA in Action”].

GPUs are graphic processing units that use the hardware-video card in addition to microprocessors–CPU to perform calculation for graphics, 3D graphic rendering and simulation applications. Because of the nature of computer graphics programs, GPUs are very efficient at calculating and manipulating parallel programs such as graphic programs, which typically have a parallel structure.

Projects that were created that combined GPU technology and visualization with scientific and information research are aquatic-GeoFish [Vance et al., 2009], medical-BartSim [Feibush et. al, 1999], and industrial-SeismicCity [“CUDA in Action”], OptiTex [OptiTex], and General Mills [“CUDA in Action”] and military-Joint Operations Visualization Environment (JOVE) [Feibush et. al, 1999].

SeismicCity, which uses GPU and CUDA technology, was created to improve the chances of finding oil [“CUDA in Action”]. The cost of drilling oil wells is expensive, so

researchers and programmers devise a program that was cost-effective. Drilling wells is estimated to cost hundreds of millions of dollars. Incorporating depth imaging technology that interprets seismic data leads to the faster selection of drilling location-in essence save both time and cost for performing the procedure.

General Mills used simulated data and visualization techniques to optimize their technique of cooking frozen pizza in the microwave ["CUDA in Action"]. The idea was to discover the effects of microwave radiation on various permutations of pizza elements rather than physically creating the different combinations. General Mills found it less expensive and time-consuming to model the pizzas using sophisticated computers.

BartSim is a program that produces high quality images of macromolecules in real-time. JOVE is used by top level military decision makers for situational awareness. Situational awareness visualization applications require the representation of large geographic areas and thousands of military units [Feibush et. al, 1999]. Visualizing the area and creating scenarios is highly useful when devising strategies that will aid in the prevention and combat of enemies attacks.

Currently there are no documented applications that combine GPU technology and advance graphics cards with stream restoration planning. However, the documented projects have been shown to use this technology for rendering terrain for virtual environments. This lack of using the technology may be the expense and the computational intensity involved in implementing, storing, and processing these applications using GPU technology. In essence, this technology does show promise for this area of study (stream restoration) because it is used for virtual and simulation environments.

The last method that increases or enhances numerical models with graphical and visualization capabilities is using gaming engine technology. Tools or applications using gaming engine technology are in areas such as weather forecasting, terrain rendering, virtual reality environment for military strategy assessment and planning, search and rescue planning, real-time environmental and interaction infrastructures, architectural design and planning, pollutant and environmental simulation and visualization, among many other areas. A few applications created that have benefited from the interactive capability and visualization features of gaming engine technology are applications such as SIMWIZ3D [Wegner, 2005], GENETICS [Wells, 2005], and USARSIMS [Roberts et. al, 2008].

SIMWIZ3D was created by Katja Wegner at the Bioinformatics and Computation Biochemistry EML Research facility using gaming engine technology [Wegner, 2005]. SIMWIZ3D is a novel software tool produced to visualize the results of data simulated from living cell life processes. SimWiz3D is an extension or expansion of the previous software SIMWIZ. SIMWIZ3D overcome many of the limitations of SIMWIZ. By using gaming engine technology, SIMWIZ3D provided the users of the software with many benefits. SIMWIZ3D provided users with the ability to visualize continuous time series data. SimWiz3D provided users with better interaction with the application. The tool has the capability to be customized based on the demands of the users. The tool can also be used to explore local and global data views. SIMWIZ3D uses small and large datasets with a large number of reactant and time steps, a major difference from existing models and the older model SIMWIZ. In addition, SIMWIZ3Ds display of the results of the simulated data has been found to be clearer and more concise. As a result of these benefits, the software usability and understanding of the process of

genetic reactants and living cell networking processes has increased.

Generating Enhanced Natural Environments and Terrain for Interactive Combat Simulations (GENETICS) is another tool using gaming engine technology, developed by Major William D. Wells at the MOVES Institute Naval Postgraduate School [Wells, 2005]. GENECTICS is a downloadable SOARXTerrain component of the military's open source game engine DELT3D. Used in combat operations this application/tool provides military authorities with the ability to devise strategic plans based on different scenarios in a realistic simulated training environment.

The CRYTEC game engine FarCry, is the gaming engine used to develop the Grange Gorman hospital site [Mcatmney et. al]. The purpose of using a game engine is to provide the developer with consultation and planning scenarios for dynamic and efficient walkthrough with interaction capabilities of the space. Using Farcry to render the hospitals terrain topography so that objects in the scene will be displayed truthfully within the environment is the primary benefit of using CRYTEC gaming engine.

Another tool created based on gaming engine technology was developed by Cheng et al. at Nanjing Normal University in China. This application combines virtual toolkit (VTK) and the contaminant dispersion simulation model to visualize the dynamic contaminant process that occurs within waterways. This application provides environmentalists with the information needed to devise comprehensive solutions and treatment to the Taihu Lakes [Cheng et al., 2010].

The last tool discussed is USARSIMS. This tool was first developed in 2002 at Carnegie Mellon University and later released to the National Institute for Standard Technology (NIST) in 2005 [Roberts et al., 2008]. USARSims is a high-fidelity, physic-based simulation environment

for robotic development. The primary objective of this tool is to provide a simulated environment with realistic terrains and object placement for a simulated complex search and rescue environment in disaster areas to help devise plans to recovery lives in these areas. This application uses the popular game engine UnReal to render realistic scenes used in the simulated search and rescue environment.

For this dissertation, gaming engine technology will be used to enhance a model used to simulate pollutants within streams and channels to help devise stream restoration plans for the affected area. The model that will be enhanced using gaming engine technology is CONCEPTS. Likewise, for GPU technology, there are no documented applications created using gaming engine technology, especially in the area of stream restoration. With the many benefits gaming technology provides, this technology shows great promise in this area of research and science. Besides the low cost, visualization and the advanced graphics capabilities associated with many innovative gaming engine applications many would still ask, “Why use a game engine?”

So, why use gaming engines? There exist a variety of reasons gaming engines are chosen to enhance existing software applications and numerical models. These considerations include economical, graphic capabilities, software development efforts and productivity, and visualization efficiencies. By definition, gaming engines provide many components needed to create innovative tools using the latest in graphic and visualization technology. Gaming engines are complex software systems designed for creating and developing video games. The main functionality of the gaming engine is the rendering engine for 2D and 3D graphics. According to Lewis and Jacobson, the only way to have the fastest, most realistic simulation and sophisticated graphics is to trade down from the expensive hardware to basic PCs running game software

[Lewis and Jacobson, 2002]. Before gaming engines were written as software many graphics were produced from the standard CPU, video card and VRAM components. This method not only produced mediocre graphics but also the frames per seconds were slow, and the image quality or models produced were not realistic.

When realistic graphics are produced, several factors must be taken into consideration. The amount of data to be rendered and the amount of processing power provided by the CPU, the GPU and the memory available to each play a major role in the quality of the graphical models created. Therefore, many scientists, researchers, and developers are using graphics processing units, video buffering, along with the CPU to offload the computational expenses of graphics rendering. As this work is being done, graphical rendering is becoming more and more the responsibility of the video card. As a result of these changes, gaming engines, which are designed to exploit these features, can be used for serious analysis and computing and not just for game play [Lewis and Jacobson, 2002]. Indeed, the game industry is actually pushing the development of visualization hardware.

Gaming engines can be used for both serious analysis and computing. The engine is considered the core of any game. Gaming engines control many aspects of a game. Most of the earlier gaming engines were very simple to develop and program. Gaming engines initially consisted of a loop event, state tables, and graphics routines. Modern games engines are far more complex. Today gaming engine are developed to controls the players' movement, navigation, light, visual, sound, screen resolution, and objects displayed. When creating a gaming engine several associated components must be also created.

From a software engineering perspective, the use of the gaming engine provides many

benefits. Gaming engines not only increase productivity when creating 2D and 3D graphics, but also they promote code reuse, thus reducing software bugs and limitations. They also provide better, faster graphics. Components of gaming engines include the system, the data/tools, the console, support, the rendering tools, and the game interface.

- Tools, as defined by Hodorowicz, at the very least, include 3D model editors, level editors, and graphics programs. [Hodorowicz, 2006]

- The system consists of sub-system such as graphics, input, sound, timer, and configuration to communicate with the machine. Systems are responsible for initializing, updating, and shutting down the sub-system. Each sub-system of the gaming engine plays a vital role in the realism and graphic capability of a virtual world's graphics.

- The graphics sub-system deals with the screen graphics

- The input component manages data sources from the keyboard, mouse, game-pad and joystick interfaces.

- The sound system is responsible for loading and playing sounds in the virtual world.

- The timer is responsible for time management in real-time gaming environments.

- The configuration subsystem enables users to customize the system.

- The console component of a gaming engine is used for testing and debugging the system during the system development phase.

- Support consists of algorithms, memory managers, loaders, and containers.

- The renderer with its sub-system is responsible for rendering the 3D graphics. [Hodorowicz, 2006]

Each component of the gaming engine is vitally important to the realism displayed in an interactive, virtual reality environment.

Economically, the use of gaming engines is ideal because of the reduced cost associated with creating tools that provides visualization components. There exist different levels for each gaming engine. Gaming engines are in the form of proprietary engines, freeware, and open source rendering engines. Each has its own benefits. Proprietary gaming engines claim to provide users with stability, user and customer support, as well as expert business solutions. Freeware, although not free, provides users with a tool that could be bought at a small cost, but with some legal restrictions on using the software. The open source gaming engines are free, and they give the users benefits that proprietary engines provide but also many benefits that freeware does not. Open source gaming engines have the benefit that it costs little to nothing as well as allows for customizable applications to be developed. The source code for these gaming engines is available to be modified and enhanced for a true customize application. Open source software is a better option for a developer working on a tight budget but, who wants the capability of creating a customizable, innovative tool that uses the latest in visualization and graphic technology.

For this dissertation open source game engines were chosen because of the major budget conscious aspect associated with creating a stream restoration planning tool and the graphic and visualization features it provides. The cost of stream restoration planning costs billions of dollars. These costs have continued to increase over the past 30 years [Bernhardt et al., 2007].

The overhead related to the stream restoration projects are staggering. On average, according to Palmer et al, the cost for river or stream restoration efforts in the United States is more than one billion dollars a year [Palmer et al., 2005]. It has been found that this cost is only associated with the requirements needed for assessing the circumstance or the current state of the

stream or river and the implementation method that will be used to restore the stream or river. This cost does not include future analysis of the river after the plan has been implemented. A problem that exists for restoration projects is that they do not budget for post-assessment of the plans' implementation. So creating an economical application that provides graphics and visually enhanced tools that can be used for stream restoration planning and monitoring are ideal. Again, the tool or software created for this dissertation is created using a free, open source game engine and other sources to keep the cost of production down. The implication defined by creating the software that uses open source gaming engine technology is defined within this dissertation.

Developing Computational Models with Visualization

Depending on the problem size, simulating, and reviewing the results of numerical models is known to be time consuming. From a software developer's perspective, computational models have to overcome the common pitfalls of time and space constraints. Time constraints relate to the amount of time required for a program to execute. Space constraints relate to the amount of memory the executing program requires. For this dissertation project the simulation model, CONCEPTS, and a graphics application were combined. Both applications are very demanding computationally, requiring substantial time and memory storage.

For this project, the graphics application components must be written efficiently for managing and rendering large terrain datasets. The dataset for CONCEPTS is both large and dense. Algorithms for these problems are computationally intensive, and require developers to create efficient programs that will optimize the performance of any CPU. For software engineers, creating a program of this nature is challenging, especially when keeping time and space issues in the forefront. To alleviate these constraints programmers, scientists, and researchers are

looking into GIS, GPU, and gaming engine technology. Each technology have a common or underlying similarity, each adds visualization features to enhance the usability and clarity of the interpreted data rendered from these innovative tools.

Research shows visual stimuli provides users with a better perception of the simulated scenarios given by computational models [Rhyne, 2007]. Scientific and information visualization have been used for many years in the fore mentioned areas- education, military, industrial, and scientific research. Visualization has been used since the beginning of time for communicating. Visualization based on computer graphics and interactive simulation techniques was formally defined 20 years ago. Visualization involves large displays and stereoscopic environments that engulf the viewers in the examination and exploration process [Rhyne, 2007]. Visualization is defined as any technique for creating images, diagrams, or animation to convey and communicate a message. In 2004, visualization was classified as three different, but overlapping, areas scientific, information, and analytical visualization.

Scientific visualization is the process of transforming data into sensory stimuli, usually images [Schroeder et. al., 2004]. Scientific visualization uses interactive, sensory representations, typically visual, of abstract data to reinforce cognition, hypothesis building, and reasoning. Scientific visualization usually involves the transformation, selection or representation of data from simulations or experiments. Representing data this way allows the exploration, analysis, and understanding of the data. Scientific visualization produces visual displays of spatial data associated with scientific processes, such as the bonding of molecules in computational chemistry. Others areas that involves visualization scientific data are flow, medical, and chemical visualization. .

Information visualization is the visualizing of non-spatial data. Information visualization is catered for addressing community planning scenarios that combine diverse data sets from geographic information system (GIS), visual impact assessments, and transportation analyses [Rhyne, 2007].

Analytical visualization as defined in the report "Illuminating the Path: The Research and Development Agenda for Visual Analytics," states that, "Visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces" [Rhyne, 2007]). Analytical visualization is developed to assist in emergency responses and rapid evaluation of transportation. This area of visualization focuses on human interaction with visualization systems as part of a larger process of data analysis. Visual analytical research concentrates on support for perceptual and cognitive operations that enables the users to detect the expected and discover the unexpected.

Today there are other areas of visualization: volume visualization, education visualization, knowledge visualization, product visualization, and visual communication. Each field focuses on simulating and visualizing information so that important decisions can be evaluated and implemented. Research shows that using visualization tools has improved the decision-making process by showing analysis information that will save users effort, time, and money [Henderson and Mason, 2010].

The visualization area for the work done here is scientific. The results produced from the CONCEPTS simulator will be graphically displayed in a 3D virtual reality environment. Visualizing the results of the simulator will aid in the exploration and understanding of different stream restoration scenarios. As a result, researchers can choose a restoration plan that can be

"seen" to be most successful.

Another key component for rendering a realistic 3D model is the data supplied to the game/rendering engine. Data poses a challenge to both the rendering engines as well as to the central processing unit (CPU). As in all areas of computer science, data management in a gaming engine is a balancing act. Although the more data supplied to the engine the cleaner, sharper, and more realistic the 3D models appears, but large amounts of data pose problems for the CPU and graphics buffer, tending to slow the whole project down.

Besides the data supplied to the rendering engine, other factors that affect the effectiveness of a tool is the programming language used to create or develop the software. Languages that closely resemble machine language are more effective for graphics and hardware manipulation than those higher level languages that do not take into consideration memory and systems obstacles. Choosing the appropriate language can alleviate many daunting issues relating to speed (time) and memory (space). Programming languages have been found to simplify and shorten software development time. The appropriate programming language provides a developer with not only a high-level, easy to use language, but, also data structures, and memory management features. These attributes allow developers to produce programs that will optimize CPU performance.

Game Engines and Programming Languages

The languages considered to be used to write the additional components used along with a gaming engine for this research project were all are domain-specific. Special-purpose or domain-specific languages provide two main advantages over general-purpose languages. One advantage is domain specific languages increase productivity because of their higher level of

abstraction. Two, domain-specific languages allow developers to express ideas in a notation familiar to the developer. Many domain-specific languages exist. Domain-specific languages are categorized as string-manipulating languages, list-processing languages, simulation languages, scripting languages, and graphics programming languages.

The languages discussed in detail for this dissertation are the most popular graphic application languages or graphic programming languages. The choices for application programming interfaces that support the special purpose languages are: OpenGL and DirectX/Direct3D. Languages are categorized as either freestanding or embedded. Freestanding graphic programming languages are languages that do not require support from other languages such as virtual reality modeling language [VRML]. Embedded languages are those that require other language support such as C, C++, Java, Fran, and Cg. A discussion of OpenGL with C++ and OpenGL with Java is provided. DirectX application programming and interface will be discussed in this section. The advantages and disadvantages of each language will be provided.

Freestanding Language: VRML

VRML was created and released by Silicon Graphic Inc (SGI) in 1995 [Yee]. VRML is a 3D graphic programming language that allows developers to create 3D scenes that can be rendered and manipulated through a web browser. VRML is interpreted by a VRML browser. VRML is one of the languages considered to be the international standard for 3D graphics modeling. VRML is maintained by the International Organization for Standardization (ISO).

The language provides developers with a high-level of abstraction to describe and create 3D scenes. The data structures that VRML uses to organize the objects or scene render is a hierarchical graph. The object vertices in the graphs are called nodes. The scene in VRML is

rendered by traversing the graph, performing a breadth-first search. The nodes of the graph represent the geometry/shape, property-texture, lighting, color, and material are combined node in a 3D scene. VRML provides developers with simple and complex 3D primitives to render intricate scenes.

The designers of VRML had several goals in mind when creating the programming language.

1. VRML was created to unite and standardize the rendering of 3D graphics over the Web.
2. VRML was designed to be backward compatible.
3. The language is scalable to support both complex and simple objects and scenes.
4. VRML objects are reusable.
5. VRML allows multi-user interaction over the web.
6. VRML language is extensible.
7. VRML is portable to various platforms.

One drawback mentioned by Yee in the article "A Survey of Graphics Programming Language" is performance. This problem is mainly due to rendering of a scene via the Internet [Yee]. In the worst-case when scenes are rendered the portability and performance will become slow during high traffic.

Embedded Languages or APIs: DirectX and OpenGL

DirectX created by Microsoft is a suite of technologies that offers developers an API to implement and design Windows-based applications. The suite used for creating 3D graphics is called Direct3D. Direct3D is a high-level modeling language with low-level APIs to access

graphics hardware. With Direct3D, programmers can compose high-performing graphics on any Windows-based system. With Direct3D, programmers have access to the sound card, memory, input devices and networking interface of Windows hardware.

When Microsoft created the DirectX suite, they designed it according to the same design objective as OpenGL and VRML. Direct3D has the features and capabilities listed below:

1. Direct3D is portable.
2. The library is backward compatible.
3. Direct3D is extensible.
4. Direct3D is easy to use.

The drawback of using DirectX is that it does not support the use of any other operating system (Linux, UNIX, and Mac OS) besides Windows. The Direct3D API does not support the Java programming language. Direct3D is a proprietary API.

OpenGL

OpenGL, developed in 1992 by Silicon Graphic Inc (SGI), is a cross-platform, cross-language API that allows developers to write 2D and 3D graphics applications [McReynolds and Blythe, 2005]. The API consists of over 250 function calls used to render complex 3D scenes from simple primitives ["OpenGL: The Industry's Foundation for High Performance Graphics", 2000].

OpenGL is widely used in computer aided design (CAD), virtual reality, scientific visualization, information visualization, and simulation applications [“3D Graphics”, 2011]. To support this wide array of applications, OpenGL was designed to support different operating systems and to support different languages. OpenGL supports Mac OS, UNIX, Linux, and

Windows operating systems. OpenGL provides an API for C, C++, Java, and Ada programming languages. By supporting different languages and operating systems, OpenGL has become a popular environment for designing and implementing 2D and 3D computer graphic applications for the web.

Other design capabilities that OpenGL possesses are as follows:

1. OpenGL is a sequential API. New processes only start when the previous process has finished.
2. OpenGL supports many operating systems and programming languages.
3. OpenGL binds objects on call.
4. OpenGL is object-oriented.
5. OpenGL is highly portable.
6. OpenGL can be executed on many system architectures from supercomputers to small portable devices such as cell phones.
7. OpenGL provides an API to interface with new technology, GPUs.
8. OpenGL is scalable and extensible.
9. OpenGL is well documented and supported.
10. OpenGL is open-source.

The language/graphic library combination that will be taken into consideration for this dissertation is OpenGL with Java, C++, and C#. These two languages/APIs are the most practiced, supported, and documented environments for creating web-based graphic applications. Java and C++ alone also provide programmers with extensive capabilities.

Java

The Java programming language first developed to create programs for handheld devices

and it has become increasingly popular for developing small games that run on mobile devices [Wilson and Clark, 2001]. Java is considered to be a small language that achieves much of its power through an extensive and comprehensive standard library, the Java API. Java API provides developers with many predefined components to construct programs quickly. The major feature of Java is that the same program that executes on one user machine can be executed on many machines with different platforms using the concept of a virtual machine. Java is a high-level, object-oriented language that is known to be platform-independent, portable, and secure. Although there are many advantages to the Java programming language and APIs there are some major drawbacks when developing gaming engines.

The Java programming language is not considered a huge contender for programming games engines for several reasons. The reasons include:

1. Java is considered too slow for game engine programming
2. Java has been found to have memory leaks.
3. Java is not supported on game consoles.
4. Java automatic garbage collection feature often runs during inconvenient times and detracts from the visualization process.
5. Java is considered too high-level for producing graphics application/gaming engines.

For these many reasons the Java programming language was not chosen for this research project [Sangappa et al., 2002].

Java with OpenGL

Java with OpenGL is the language used today for developing rendering engines for graphics applications [Wolff] [Day]. Java OpenGL or JOGL, developed by Kenneth B. Russell

and Christopher J. Kline, is a wrapper library that allows OpenGL to be used with the Java programming language. After further development of the language, by Sun Microsoft Game Technology Group, JOGL is available as an open source project under the BSD license. Many benefits of using this particular programming language are established. This language is one of the core projects of the Java gaming community at the Sun-sponsored community website (Java.NET, 2010). The project is under active development.

Many graphics application developers claim that Java is not a serious programming language for developing serious graphics applications or gaming engines. The drawback of using JOGL is that developers find that JOGL has limited technical support; it contains many bugs, lacks documentation, and is incomplete when compared with C++ with OpenGL. An alternative language that claims to alleviate many of the stated problems is the language called GL4Java. GL4Java developed by Jausoft, works with the latest version of OpenGL, Graphic Library Utility (GLU), and the Graphic Library Utility Toolkit (GLUT) [Davison, 2005].

C/C++

C developed by Ritchie and Thompson at Bell Laboratories is a systems programming language [Wilson and Clark, 2001]. C was developed and used to implement the UNIX operation system. C has several benefits. C provides developers with the advantages of a high-level language with the facilities and efficiency of an assembly language. The C++ language, later developed by Stroustrups, is the object-oriented extension to C. The advantages of C++ include [Wilson & Clark, 2001]:

1. C++ is safer than C.
2. C++ supports low-level system programming, which is highly vital for graphics programming.

3. C++ supports object-oriented programming. C++ is highly portable.
4. C++ is efficient.
5. C++ compiler is available on any platform.
6. C++ is the most widely used programming language for developing gaming engines.

The drawbacks of the C++ language are few. C++ is not considered a purely object-oriented programming language, unlike Java. C++ uses pointer-based arithmetic, which could be a challenge for even the most proficient software developer.

C++ with OpenGL

C++ with OpenGL is the preferred programming language environment for developing graphic applications [Glaeser and Hellmuth, 1999]. Because of the fact that OpenGL was written in C++, it makes OpenGL with C++ programming highly efficient. Therefore, the language that will be used for this project is a variation of C++.

.NET C# with OpenGL

The .NET framework also created by Microsoft is a platform that provides developers with the tools needed to create applications quickly and easily [Miller, 2011]. The .NET framework consists of many features and mechanisms. The .NET mechanism tools include memory management, code management, a runtime environment, common language runtime, and four popular programming languages. .NET features provide a well understood programming model and a common set of APIs that allow developers to a create more domain specific applications. For graphics applications .NET provides the Direct3D API. The language common for graphics application is C#.

C#, similar to C++, has many advantages. C# syntax is very similar to C++ syntax but does not use pointers. The language also provides a memory management mechanism that handles memory allocation and leaks. The language also has automatic garbage collection a very popular feature of Java. C# is also a strongly typed language, a feature that reduces coding errors. The language has a built-in compiler and debugger. An application written in C# has the capabilities that upon deployment it comes equipped with the mechanisms needed to function properly. Because of these aforementioned features and mechanisms created for C# this is the language of choice for this dissertation [Miller, 2005].

Gaming Engines

A major component of the work done for this dissertation is the gaming engine interface that allows users to navigate and interact with CONCEPTS data in a 3D environment. Because of the time constraints for this research project only open source gaming engines were considered. Using an open-source gaming engine promotes code reuse, increases software development productivity and overcomes the many proprietary battles that might otherwise occur.

Many open-source, as well as proprietary, gaming engines exist. For this project only free and open source gaming engines are reviewed and considered. After significant review, the gaming engines that show most promising are Unity [“UNITY”], DimensioneX [“DimensioneX”], Cafu [“Cafu Engine”], Panda3D [“Panda3D], Delta3D [GeekLog], NeoAxis [NeoAxis Group, 2010] and OpenSimulator [“OpenSim”]. These free and open source gaming engines are considered for several reasons.

1. They allow games to be created and executed on the web.
2. These gaming engines are free and open source.
3. These gaming engines have a large support group.
4. Tutorials and documentation are available.

A comparison chart is given and each is reviewed in detail in figure 3 below.

GAME ENGINE	WEB-BASED	TERRAIN RENDER	OPEN SOURCE	LICENSE	PRICE	TUTORIALS/WELL DOCUMENTED
DIMENSIONEX (DIMX)	YES	NO	FREE-WARE	GPL	Contact DIMX	NO
UNITY	YES	NO	FREE-WARE	Noncommercial, Indie, Commercial	$1200	Yes
PANDA	YES	NO	YES	BSD	N/A	YES
DELTA3D	YES	YES	YES	LGNU	N/A	YES
CAFU	YES	NO	FREE-WARE	GPL	Contact CAFU	NO
NEOAXIS	YES	YES	YES	Noncommercial, Indie, Commercial	Contact NEOAXIS	YES
OPENSIM	YES	YES	YES	BSD	N/A	YES

Figure 3: Comparison Chart of Game Engines Evaluated

DimensioneX

DimensioneX is a JavaServer, multiplayer gaming engine that allow developers to create games by code reuse [“DimensionX]. DimensioneX runs on many platforms including Windows, Linux, and Mac. Any computer that has the capability of running Java can execute DimensioneX. This gaming engine is available for free and is distributed under the GNU General Public License.

Games created with DimensioneX are played via the web. DimensioneX allows game scenes and sounds to be displayed. The gaming engine allows developers to customize player’s avatars. The gaming engine also provides developers with the capability to allow movement and navigation of the avatars in a virtual reality environment.

Cafu

Cafu, developed by Carsten Fuchs, is a 3D graphic engine and game development kit [Carsten Fuchs Software, 2010] Cafu's source code is available for free under the GNU Public License. Features that Cafu provides that make this gaming engine a contender among others are the following:

- High-quality 3D computer graphics
- High portability
- Ability for games to be executed over a computer network or Local Area Network (LAN)
- Large terrain renderer for developers
- Free to use and modify ["Cafu Engine"].

Panda3D

Another gaming engine considered for this project is Panda3D ["Panda3D-Free 3D Gaming engine"]. Panda3D is available for free as open source under the Berkeley Software Distribution (BSD licensed). Panda3D features include the following [Panda3D]:

- Automatic graphic rendering techniques.
- Performance monitoring, optimization, and debugging tools.
- Execution on the web.
- Well documented and has available tutorial online.
- An active community.
- The capability to use the graphics API DirectX and OpenGL.

- Source code available for free [“Panda3D”].

A minor drawback of this gaming engine is the programming language. Developers are required to write in the programming language Python. Since much of the provision work done for this project was done in Java and C#, a new language seemed inappropriate at this time.

Delta3D

Delta3D is an open source game and simulation engine. It provides many of the same capabilities and functions as the other gaming engines. Developers are provided with the following [GeekLog]:

- GNU Lesser General Public Licensing making it free and open source
- Ability to be executed on many platforms such as Mac, Windows, and Linux
- A complete 3D editor
- Access to the Python API
- A client/server architecture that gives it capabilities to be executed on the web
- A framework for implementing and rendering terrains

For the listed reasons Delta3D is considered as a potential engine to be used to meet the objectives of this project.

Unity

Unity is an integrated authoring tool for creating 3D video games or other interactive content such as architectural visualizations or real-time 3D animation [“Unity”] [Craighead et al.]. The most prominent features of Unity that makes it highly desirable for this research project are the game editor, web player, graphic engine, audio system, and a terrain and vegetation engine.

NeoAxis

Neoaxis created by the Neoaxis Group is an integrated development environment and gaming engine that allows developers to create games, simulations, and visualization tools [NeoAxis Group, 2010] [. Similarily to the aforementioned gaming engines, Neoaxis implements many features that promote fast development and graphic rendering. Neoaxis provides developers with the following capabilities [NeoAxis Group, 2010]:

- Neoaxis provides a set of visual tools that enable the developer to build game level maps, interactive capability, and management of the 2D interfaces of the game.
- Neoaxis provides a height-map based landscape editor.
- Neoaxis provides a GUI editor for creating GUI Editor that allows developer to create end user controls, menus, dialogs, windows, screens, and an in game 3D GUI.
- Neoaxis also provides a terrain editor as a landscaping design tool. It supports geometry and painting alpha layers onto the terrain to control blending collision data and support of detail and normal maps.
- From the programming perspective with Neoaxis there is no compile time. Runtime simulation is instantaneous to allow interaction within the gaming environment and maps.
- Neoaxis supports 3D packages from different 3D software modeling packages such as 3D Studio Max, Maya, and Blender.
- Neoaxis can be deployed within all major web browsers.

Neoaxis has many more features that make it a highly favorable gaming engine to

consider. One of the most important features that Neoaxis has is its ability to be integrated into other applications. This feature is the main feature that made this a great gaming engine candidate. Neoaxis allows ease of integration into application frameworks for the creation of windows application. It uses WinForms integration as well as providing 3D widgets that are useful when creating simulation, graphics, and visualization tools to develop web applications projects that are quick and seamless. Neoaxis also uses the powerful .NET based API scripting for any .NET language support. It supports rapid development by promoting decoupling techniques and code reuse [NeoAxis Group, 2010]. For these many reasons this engine was highly considered as a potential candidate for this project.

OpenSimulator

OpenSimulator (OpenSim) was the last gaming engine evaluated for this dissertation. In 2007, Linden Lab open sourced the client software program. Then in 2009 they released the server program thus creating the open source version of SecondLife, OpenSim [Fiskwick, 2009]. OpenSim is a multi-users and multi-platform 3D web application server. It allows development of virtual environments to customized worlds based on technology that is easy for them.

OpenSim is easily extensible because of the basic approach taken when the application was developed. Like Neoaxis, OpenSim is written in C# and has the capability to run on Windows and Unix-like machines through the use of the .NET framework and Mono Runtime framework. The better feature of OpenSim is that it is truly an open source engine provided by Berkley Standard Development (BSD). This license allows developer to create seamlessly application by providing an easy way to integrate other modules plug-in to the application with source code at their disposal. Because of the many features that allow it to be extended OpenSim

was the game engine of choice for this dissertation.

Game Engines Evaluated and Tested

The first gaming engine that was used to render or draw the results of the output/height-map was Unity. As mentioned before, the Unity gaming engine was chosen for several reasons. Unity runs on many platforms such as handheld devices, mobile devices, and PCs. It can be executed on many operating systems. The main reason Unity was initially considered for this project is that it can be executed on the web.

Although Unity provides many features such as an editor, basic terrain environments and vegetation libraries, it had to be modified for this project. The component or code that had to be added to Unity is the capability to allow users to create and accept other forms of terrain data. The input component created allows the user to create a height-map, which is used as the gaming engines input data to render user specified terrain.

Although the Unity gaming engine provided many features, it had to be abandoned for this project. There were costs associated with the main components needed to create a customized terrain. Unity provides the functionality that will allow terrains to be imported and customized, but other components are not available to be modified. The free, available gaming engine, Unity, allows the user to view terrains but customization of textures, vegetation, and other objects is not allowed. Developers are not allowed access to the source code. The feature in which a user can use native libraries to enhance functionality is available with Unity Pro which, is offered at a price of $1200. Two of the main objectives of this project when choosing a gaming engine are to allow users to customize objects to be added to a scene of the 3D environment and to use free, open source code which allows modification with few stipulations

and no fees. The next gaming engine that was tested is the Cafu gaming engine [Carsten Fuchs Software, 2010].

The Cafu gaming engine was initially evaluated for many of the same reasons that Unity was evaluated. Cafu provided networking capabilities, large terrain rendering, and great documentation as well as had many tutorials, and is open source. One major issue that was found with Cafu is that it was not completely open source. Another issue was that designing a web application with the gaming engine required that the code be built around the gaming engine which leads to tightly coupled code. This makes it hard to reuse code as well as maintain, and debug code. Also the gaming engine required addition programming languages to be learned as well as additional code to be integrated to use the most important feature, the terrain rendering feature.

Another gaming engine that was used and tested was Neoaxis [NeoAxis Group, 2010]. Neoaxis is a proprietary based gaming engine. It comes with different licensing offers. The free, noncommercial open- sourced gaming engine promotes object-oriented programming, code reuse and loosely coupled programming design. NeoAxis is a windows-based system so each individual module can be built and integrated with it seamlessly [NeoAxis, 2010]. Neoaxis uses C# that is the language of choice for this dissertation. It has many of the benefits of C++ with the advantage of memory management, strong typing, and automatic garbage collection. These are only a few features of the many found to be of value. The API used was OpenGL for graphics, this allowed rendering to be programmed and provided direct access to features and programming of the graphic card. The major drawback with NeoAxis was the cost associated with developing a customized graphic application.

The gaming engine that was used for this dissertation provided many benefits that led to it being the primary gaming engine of choice for this dissertation. OpenSim lends itself well to being extended by allow module plugins. OpenSim is the open source version of the most popular virtual world SecondLife [Fishwick, 2009]. Many of the standard features of the gaming engine have the same architecture as SecondLife game engine. The physics engine and collision detection algorithms are from the well-known engine Havok [Hand] ["Havok"]. The communication protocol is the typical internet communication protocol TCP/IP. The built-in capabilities of the OpenSim gaming engine mimic that of SecondLife. The language that was used as the backend for OpenSim is C#. This only increases the probability of ease of integration with other components used for this dissertation.

Details of the implementation/use of the chosen gaming engine will be discussed in the methodology section. Each component created, used, and enhanced, in addition to the gaming engine itself, are evaluated and documented within a software development lifecycle framework section of this dissertation. Applying the SDLC process to develop CONCEPTS3D components increased the productivity and the correctness of the design of the entire integrated system.

Data and Data Processing

One of the main challenges of this dissertation as well as many other applications that use data supplied to the game engine for terrain or environment rendering is the dataset itself. Before the data can be used by the gaming engine renderer, many datasets must be created, modified, and/or enhanced. The data supplied to the gaming engine in this case comes from various sources such numerical models, satellite, DEM, and remotely sensed data, each of which poses additional challenges. For this dissertation the data was obtained from either satellite or digital elevation

model (DEM), surveys, and the CONCEPTS simulation model data.

Digital elevation models (DEM) or digital terrain models provide only the height of the ground surface being investigated. This is an issue because the data from DEM are not sufficient to create a realistic 3D object for displaying the topography of a terrain. Therefore, a data set must be created. From the cross-section data file the information was extracted about the channel that lies in the terrain. Combining the dataset about both the floodplain and the channel presented a representation of the information needed to construct a 3D model of the terrain for the gaming engine.

The problem with the DEM data is that it is in the form of an Environmental Systems Research Institute (ESRI) grid file format. This data file and format poses a challenge and requires further enhancement. The ESRI file format provides only the z value for the terrain topography. The x and y values will have to be calculated based on the z value to obtain the three coordinate values to create a 3D model. How this calculation was performed is given in detail in the implementation segment of the methodology section.

The data provided from the XML, DEM, and CONCEPTS cross-section files each provide its own obstacles. Within the CONCEPTS DEM and cross-section file there exist data in the file that represent the x, y, and z data values of the terrain and the channel being modeled. These data values are represented as easting, northing, and elevation respectfully. Both the DEM and the CONCEPTS cross-section dataset must be extracted or parsed to extract the correct information to supply to the gaming engine. How each challenge is addressed is the subject of the methodology section of this dissertation.

The techniques used to enhance the data set include interpolation and triangulation

methods. Interpolation is the method of constructing new data points within the range of a discrete set of known data points. There exist many forms of interpolation. While there are many interpolation method including piecewise constant interpolation, linear interpolation, polynomial interpolation, bilinear, trilinear interpolation, and spline interpolation, the interpolation methods used in the work done for this dissertation is bilinear interpolation. The other method that was used to enhance the discrete data sets is the triangulation method. Triangulating data is the process of determining the location of a point by measuring angles to it by known points at either end of a fixed baseline [Rui et. al.]. From the perspective of geometry, triangulation is the process of subdividing an object into triangles or tetrahedrons. The triangulation method used here is Delaunay triangulation. Details of each of the methods and how and why they are used are discussed later in the methodology section of this dissertation.

To obtain the points necessary for visualization, linear interpolation of the data points from two known points is determined. Linear interpolation is the easiest to understand and implement when choosing among several interpolation methods. Interpolated data between two points is a straight line connecting those two points. Although this method is simple to grasp, it is highly error prone especially in terrain models. Another drawback to this method is that it does not provide a smooth curve data set. Better interpolation methods to produce continuous, smooth, and curvature data set are bilinear interpolation and spline interpolation.

Spline interpolation also calculates new data values from existing data. Spline interpolation, (as it is suggested from modeling objects using the spline technique), is based on curve fitting [Ding and Rossiter, 2007]. Spline interpolation is a form of interpolation in which the interpolation is a special type of piecewise poloynomial called a spline [Grevera et al., 1998].

Spline is preferred over polynomial interpolation because the interpolation error can be made small even when using a low degree polynomial for the spline. Another benefit of spline interpolation is that it avoids the problem known as Runge's phenomenon, which occurs when interpolates between points, have equal distance with high degree polynomials.

Bilinear interpolation was the interpolation method used on the data set for this dissertation. Bilinear interpolation is an extension of linear interpolation meaning that instead of interpolation done in one direction, interpolation is done in both the x and y direction. Bilinear interpolation produces smooth and continuous data, which produces a realistic dataset for rendering in 3D visualization. Bilinear interpolation performs interpolation by using four points that are known and finds the weighted approximate value of the unknown points to provide an additional point for the discrete data set. Because bilinear interpolation is designed for rendering 3D data this was the interpolation method chosen. The final method that was used to enhance the data set supplied to the gaming engine is a triangulation technique called Delaunay triangulation.

In this case, Delaunay triangulation involves the process of subdividing a polygon (here in a 3D terrain) into triangles with the stipulation that any point in the plane does not belong within the circumsphere of any other point within the same plane. Remembering that our goal is to move from 2-D data to realistic 3D data, it can be seen that the Delaunay triangulation now provides a method for increasing the granularity of the data by further subdividing the initial triangle into even smaller triangles that meet the criteria of staying within the 2D dataset while at the same time providing more information to the rendering engine. There are many efficient algorithms that exist for Delaunay triangulation [Razafindrazaka, 2009]. The drawback of the

Delaunay triangulation algorithms is that for some cases it may not be able to produce a Delaunay triangle [Daintith , 2004]. Since the focus of this dissertation is the dimension of the data, three dimensions, producing triangles using Delaunay triangulation is not an issue. To view the object rendered by the Delaunay processing, barycentric interpolation was perform to the data.

From each of the above techniques the positive outcome of the new data set is that it produced a better image resulting in a more realistic model. The drawback was the more data produced, the slower the processing of the data as well as the increase of space required for storing the data.

For drawbacks of large datasets produced, there are many known techniques used for solving the problems related to the visualizing and modeling of larger 3D terrains [Han et. al., 2003], [Lim and Choi, 2008]. Methods such as using triangle patches and cutting triangles within a dataset have been shown to render better 3D models [Lim and Choi, 2008], [Sanders et. al, 2000]. Cutting triangles based on the level of detail (LOD) is the technique used by the gaming engine used within this dissertation. This solution is chosen because it improves the image being rendered and stored by eliminating polygons based on what the human eye and brain perceives as necessary to view a model or object correctly in 3D.

The methodology section will discuss the game engine used as well as the data enhancement techniques used to render the 3D environment from the CONCEPTS model in greater detail. Also each objective and solution for each task for developing CONCEPTS3D is given in significant detail in the methodology section.

CHAPTER III

METHODOLOGY

In this Chapter, a description of the procedures and tools used to develop the CONCEPTS3D model are discussed. Several objectives were articulated for this dissertation. The main objective was to provide the user with a tool in which the CONCEPTS model data could be displayed in a 3D virtual environment using gaming engine technology. The CONCEPTS3D desktop application allows the user to manipulate, store and access CONCEPTS data in a database. The CONCEPTS3D model also allows the user to view the topography of the terrain and channel being analyzed and modeled in 3D. The procedure and tools used to realize each objective will be discussed in detail here (See Figure 4). A map of the procedure is provided in Figure 4 on page 54.

The first objective was creating the database schema based on the XML Scheme Definition (XSD) file that provided all the data relating to CONCEPTS. This XSD file provided the mechanism to allow the viewing all of possible combinations of XML files of CONCEPTS data. The XSD file used to create the XML files provided the relationships to each CONCEPTS table in the XML file. After data was parsed from the XML file (A) the data is stored in the CONCEPTS MySQL database (G). (See figure 4)

The second objective was to create three parsers to extract information from the XML

(A), DEM (B), and cross-section (X-Section) (C) files. The DEM (B) and X-Section (C) files were parsed for 3D information about the terrain being visualized. The DEM (B) file provides information about the terrain floodplain of the associated XML (A) file. The cross-section (X-Section) (C) file contains information about the channel that lies within the terrain. Once information was extracted from the DEM (B) and X-Section (C) files, these files were merged to create an integrated topography of the terrain under investigation (I) to be visualized in 3D.

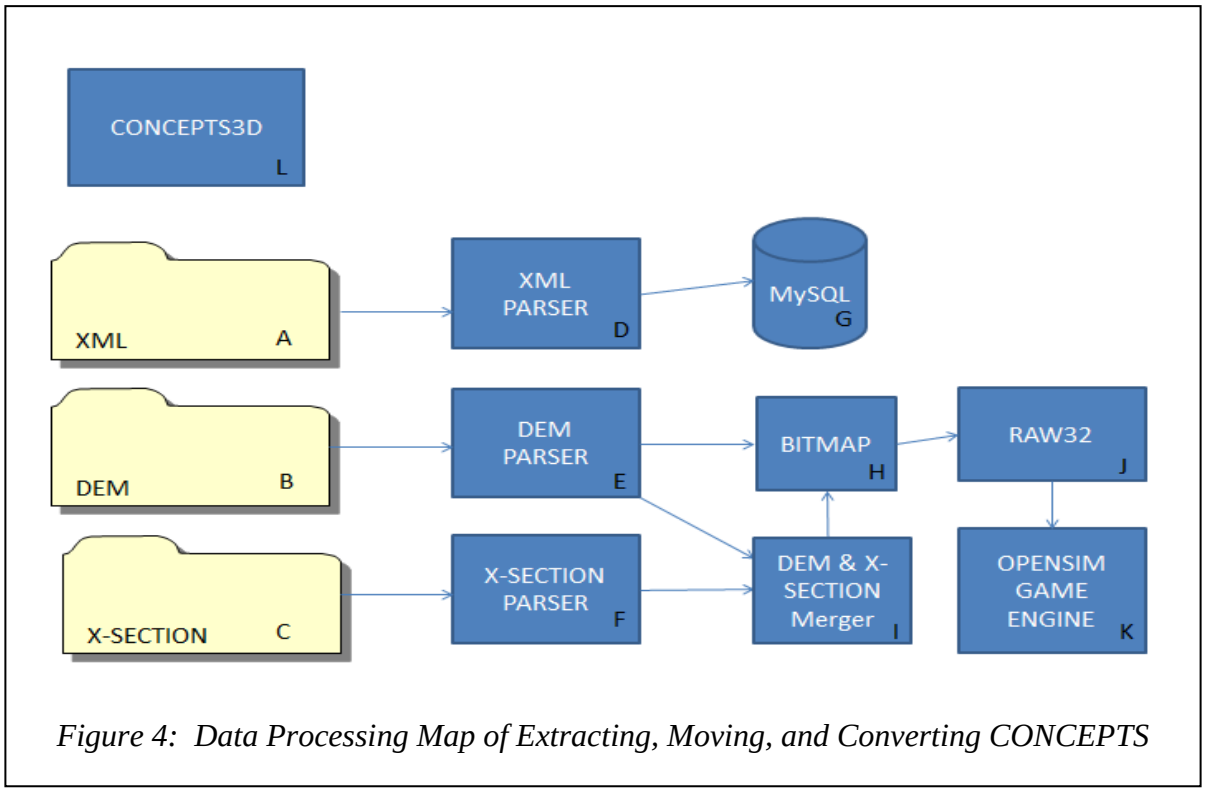


Figure 4: Data Processing Map of Extracting, Moving, and Converting CONCEPTS

The implementation details of each parser component are given in greater detail in the implementation section titled Creating Concepts3D: SDLC section.

The third objective was to use the information provided from both DEM (B) and X-Section file (C) to create a HeightMap (H) file. The bitmap file, used for displaying purposes, is

a visual representation of the HeightMap (H). The selected area of the bitmap image was used to create the native file format HeightMap RAW32 (J) file used by the gaming engine OpenSimulator (OpenSim) (K). Each objective implementation is given in detail in the implementation/code portion of this section.

The fourth objective was selecting an Open Source gaming engine that will allow the user to visualize CONCEPTS data in a 3D virtual world. Among the many gaming engine OpenSim was the engine of choice. OpenSim is the open source version of SecondLife, which is the most popular gaming engine [Fiskwick, 2009]. OpenSim was created by Linden Labs and available through BSD open source license. A few of OpenSim built-in capabilities allow navigation within the space, change of scene for clearer visual of the space, snapshot of the area being analyzed, and visualizing on-the-fly terrain updating. Many of the built in features of this gaming engine features presented by OpenSim mimics that of SecondLife.

The final objective was creating the desktop application that will allow the user to interface with the underlying components of CONCEPTS3D (E). The desktop application provides the ability to select files to visualize in the virtual world. The components are three parsers (D, E, and F), a HeightMap (H) generator, and the gaming engine (K). The parser is written using a .NET framework programming language - C#. There were also functions developed to enhance the data from CONCEPTS. As discussed earlier, these functions use interpolation and triangulation methods. After the data is enhanced, the data is saved as a HeightMap Raw32 (J) that was used by the gaming engine. The gaming engine, OpenSim, renders the HeightMap Raw32 (J) file into 3D. Some of the file formats that the game engine allows for 3D objects in OpenSim are .bmp and .raw for better quality images. Details of the

implementation process are given in the implementation/coding section under developing CONCEPTS3D software development lifecycle.

Developing CONCEPTS3D

To build a prototype of the new, innovative tool, CONCEPTS3D, there are many tools and techniques used and built to integrate the components for the new model. CONCEPTS3D allows the data of CONCEPTS XML, DEM, and cross-section files in a desktop application window panel in 3D. The steps to complete these objectives were listed as follows:

1. Components were written to extract CONCEPTS data that was modeled in 3D.
2. The data from CONCEPTS was maintained in a MySQL database. Therefore, a database schema was written to represent the CONCEPTS/CONCEPTS3D data.
3. The data set was created and enhanced using an interpolation and triangulation method.
4. The CONCEPTS model data was available for testing the CONCEPT3D application.
5. The data of CONCEPTS was parsed to get the terrain topography for the evaluated stream.
6. The two input files DEM-terrain and X-Section file were merged to create a file of the full terrain topography under investigation.
7. Code to visualize the data was written to be viewed with the OpenSim gaming engine.
8. Each component of the new CONCEPTS3D model was tested.

Each component added to the game engine was evaluated and documented according to the software development lifecycle. Applying this process to develop CONCEPTS3D

components increased productivity and insured correctness for the design of the entire integrated system.

Creating CONCEPTS3D: Software Development Lifecycle

Developing software requires programmers to follow specified steps based on some form of software engineering practices and principles. Studies show that following software development guidelines shortens the software development time, reduces coding errors, and reduces debugging time. Many software development processes and guidelines exist. For this dissertation the waterfall model was followed.

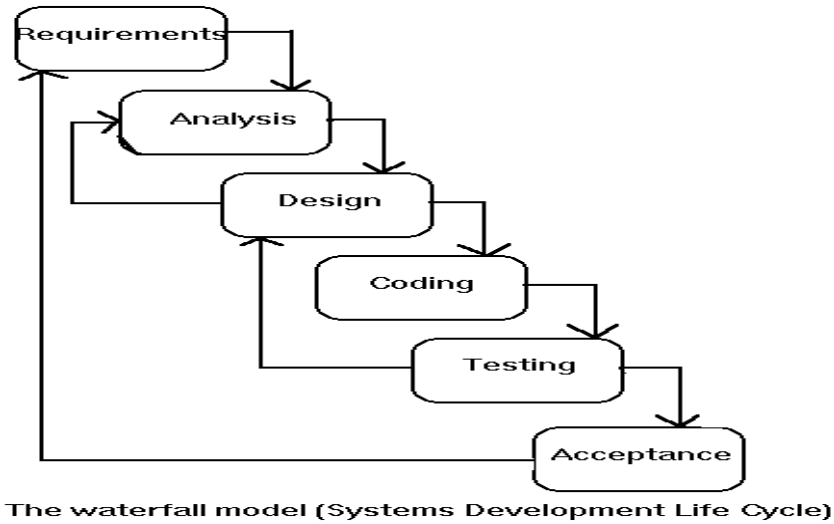


Figure 5: Waterfall SDLC Model [Gangolly, 2000]

The steps that are discussed for the development of CONCEPTS3D are listed as follows:

1. Specification and Requirement
2. Design
3. Implementation/Coding

4. Testing
5. Maintenance
6. Validation and Verification
7. Acceptance

Each of these steps is discussed in greater.

Before the software development lifecycle (SDLC) begins the developer must first understand the problem to be solved.

System engineering and Analysis

The software engineering SDLC begins. In the system engineering and analysis phase begins with the software components that interact with one another are evaluated. System engineering and analysis entails gathering requirements at a system level. The developers establish what is required for the system elements. For this dissertation research the CONCEPTS3D desktop application, data extraction components, database management system MySQL, and rendering engine module were integrated.

Components are required to run on Windows 95 or later operating systems. The amount of hard disk and memory space will depend on the size of the problem being calculated. The integrated model requires the user to have a minimum of 32 MB of RAM. To use the integrated model, the CONCEPTS3D desktop application installed on the user machine, the game engine-OpenSim will run inside of that CONCEPTS3D desktop application.

Specification/Requirements

In the second phase, specification and requirement analysis, developers focus on gathering specifications for the software. The developer must understand the domain and all

requirements for the software, the function, performance, and interfaces. Requirements for both the system and the software are documented and reviewed with the client (USDA).

Currently, CONCEPTS requires at least four input files created by the user:

1. one input file with run control data
2. one input file with discharge data at the upstream boundary of the modeling reach
3. an input file for each cross-section in the modeling reach
4. an input file for each hydraulic structure in the modeling reach

In the integrated model, the CONCEPTS numerical model data is the only information used to execute the CONCEPTS3D model. To execute CONCEPTS3D, the user is required only to provide a DEM file and an X-Section file, to visualize the terrain in 3D. Much of the conservation measure information required for CONCEPTS is specified by default for the CONCEPTS3D application, which is stored in a database. The desktop application that was created for CONCEPTS3D requires the user to provide the input file about the terrain topography. This information was parsed so that the CONCEPTS3D terrain parser component would extract the information required for 3D rendering. After the CONCEPTS3D parser component has executed, the output of the application was made into a HeightMap/bitmap file that was used as input to the game engine.

Figure 6 below shows how the raw data, DEM file (A), is transferred into a bitmap file (B), which is used by the OpenSim gaming engine (C) to render a subsection of the bitmap as a virtual reality terrain of the DEM information (See Figure 6).

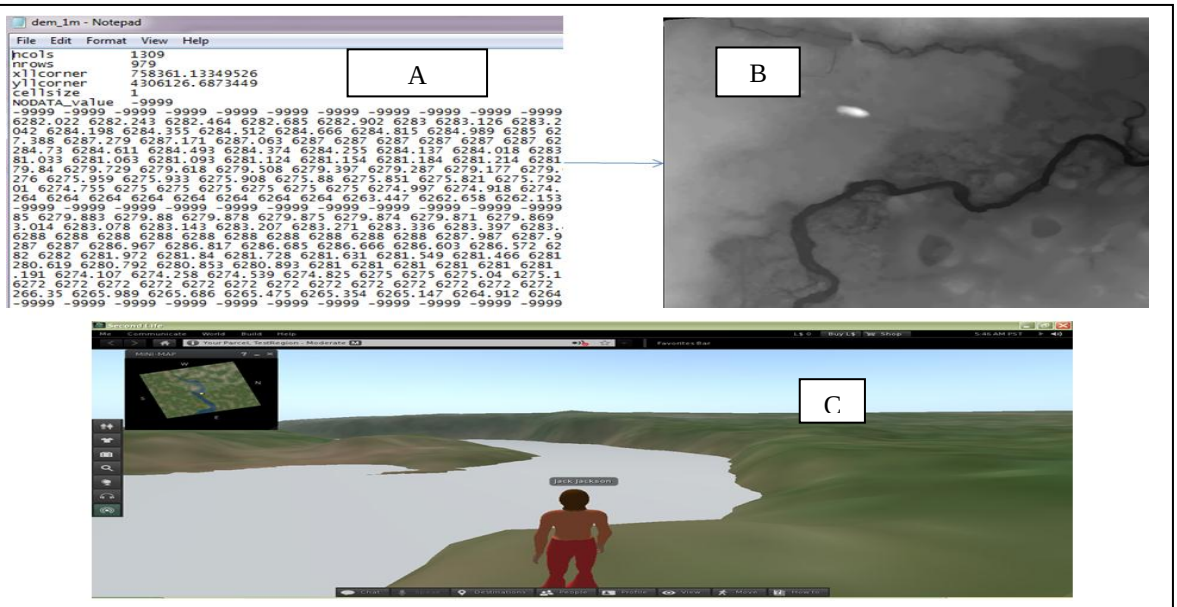


Figure 6: File format process before Virtual Environment Rendering by OpenSim

Design

The design phase is a multi-step process that focuses on four distinct attributes of a program. These four attributes are the data structures, software architecture, procedural detail, and interface characteristics. The architecture of this project is shown in the Figure 7 below.

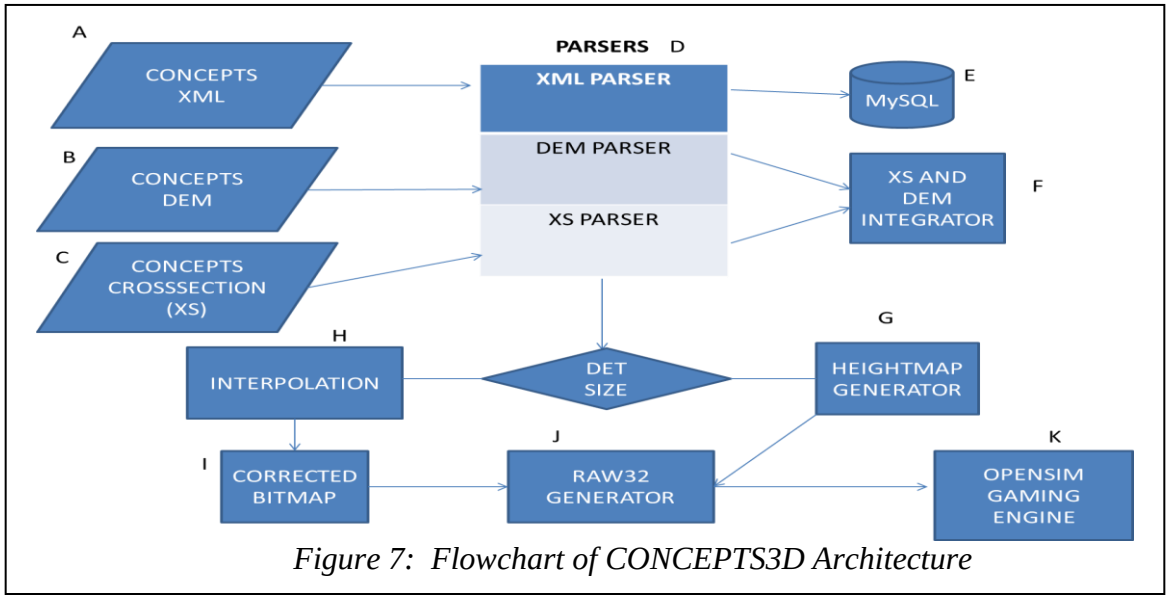


Figure 7: Flowchart of CONCEPTS3D Architecture

The information from CONCEPTS is inserted, maintained, and accessed from the MySQL database (E) (See Figure 7). The desktop application created for CONCEPTS3D allows the user to extract certain information from the DEM file (B) and cross-section file (C) - such as the terrain topography and channel topography. When executing the CONCEPTS3D model desktop application three files are used, the data files DEM (B) and X-Section (C) and XML file (A). Once the XML file is parsed by the parser component (D) the resulting XML (A) information is inserted into the MySQL database, the DEM (B) and X-Section file (C) that are the associated file of the XML (A) file are used by the parser component. The parser creates a HeightMap/bitmap file (G) from the DEM (B) and X-Section (C) files that was used as input for the OpenSim game engine (K).

Once each component was performing as expected, they were combined based on the pipeline describe above in Figure 7. The integrated system allows the user to view CONCEPTS data in a seamless virtual realty world --CONCEPTS3D. In this environment the user is allowed to interact in the 3D space, walk or navigate within the environment, and select a smaller region to investigate the topography in the larger terrain. Figure 8 below shows L of figure 4 the prototype opening screen of the entire integrated system.

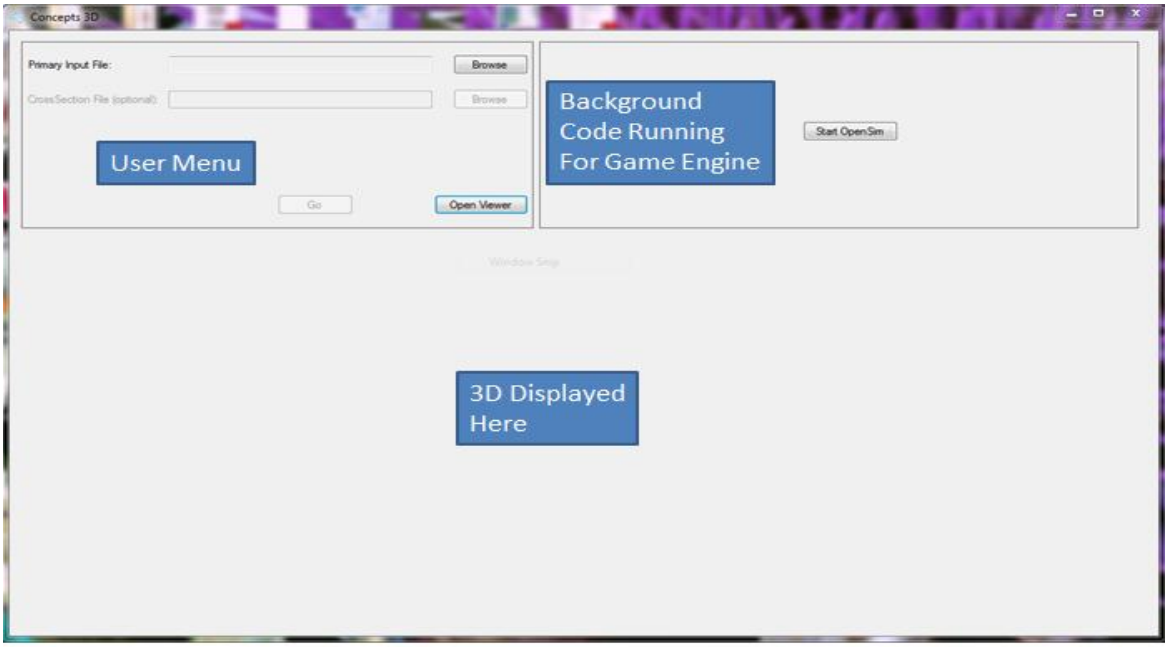


Figure 8: Prototype CONCEPTS3D Desktop Application

Implementation/Coding

In the implementation/coding phase, the algorithms developed by the software developer or programmer must be translated to machine-readable code. For this dissertation the .NET Framework is the basis for the entire infrastructure of the CONCEPTS3D application. The language used for writing the application is C#. The Graphics API used with C# for the graphics component was OpenGL. Making the entire system windows based by using the .NET framework has proven to have many benefits. One major drawback of using .NET is that it is not supported by many operating systems, thus limiting the user to using windows based machines.

Creating the application based on .NET from a software engineering perspective allows the program to be more robust, increases interoperability, permits easy integration with other

Windows applications, increases code reuse, reduces bugs, and allows for quick discovery and fix of bugs once found. Integrated components are seamlessly merged and were found to work correctly because of the common underlying infrastructure available by .NET. .NET uses the Common Runtime Language (CRL) as the underlying architecture of the .NET framework. The CRL is the virtual machine that manages code and translates code to the native machine language for execution. This component allows the user to use the .NET language of choice for components developed for any application.

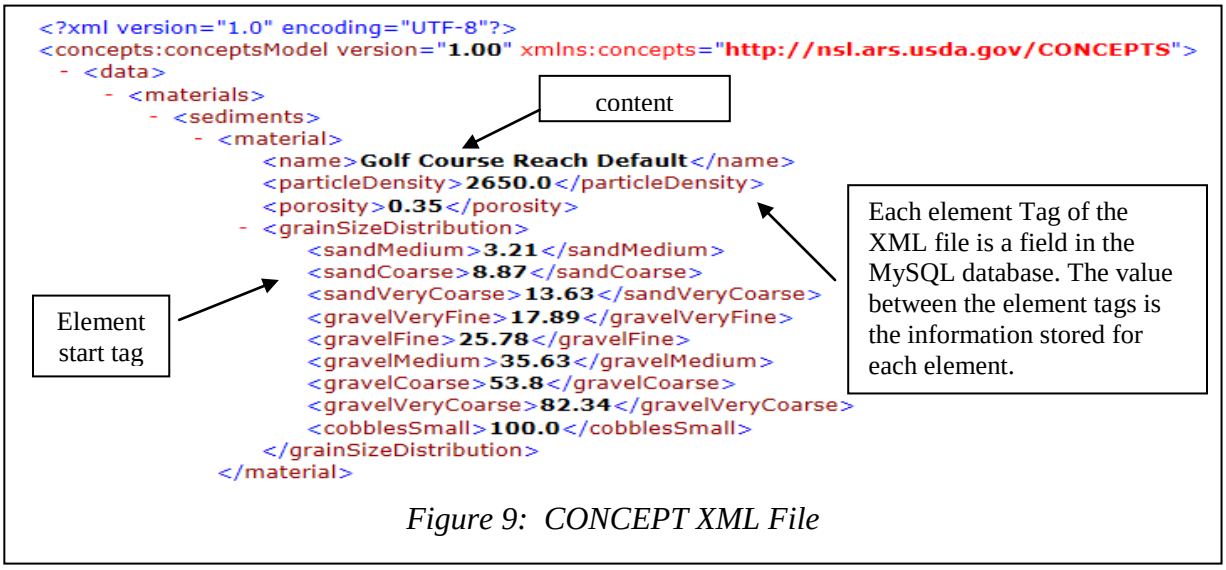
Experience shows that integrating components can be a difficult task. This is very much a factor in integrating applications with a gaming engine. This is because in many applications those integrated components are written to be tightly coupled within the application. It has been found that applications have to be hardcoded to work with the gaming engine systems correctly, thus creating a tightly coupled application. Maintaining and upgrading these systems is tedious. Bug repair is also cumbersome, inevitably requiring the entire system to be rewritten, wasting both time and money.

Developing loosely coupled applications is preferred. Separate modules that work as individual components allows the developer to focus on the logic of the application integrated with a gaming engine. This is the primary and valid reason the .NET framework was chosen as a basis for the entire CONCEPTS3D application.

Each component written used the .NET framework language. The desktop application, parser component, bitmap, and HeightMap file generators were all written in C#. The gaming engine OpenSim's underlying infrastructure is also written in C#.

In essence, there are two parsers. The first two parsers takes the DEM file and cross-

section file, then extracts the appropriate information to create a 3D model. The second parser extracts data from the CONCEPTS XML file to store into the MySQL database. The XML/XSD file has tags that describe the input information for CONCEPTS data. See figures 9 and 10 below.




```
<?xml version="1.0" encoding="UTF-8"?>
<concepts:conceptsModel version="1.00" xmlns:concepts="http://nsl.ars.usda.gov/CONCEPTS">
 - <data>
    - <materials>
       - <sediments>
          - <material>
              <name>Golf Course Reach Default</name>
              <particleDensity>2650.0</particleDensity>
              <porosity>0.35</porosity>
            - <grainSizeDistribution>
                <sandMedium>3.21</sandMedium>
                <sandCoarse>8.87</sandCoarse>
                <sandVeryCoarse>13.63</sandVeryCoarse>
                <gravelVeryFine>17.89</gravelVeryFine>
                <gravelFine>25.78</gravelFine>
                <gravelMedium>35.63</gravelMedium>
                <gravelCoarse>53.8</gravelCoarse>
                <gravelVeryCoarse>82.34</gravelVeryCoarse>
                <cobblesSmall>100.0</cobblesSmall>
              </grainSizeDistribution>
          </material>
```

Figure 9: CONCEPT XML File

```
<xsd:element name="materials">
  <xsd:complexType>
    <xsd:annotation>
      <xsd:documentation>List of bed sediments and streambank/floodplain soils.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="sediments">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="0" name="material" type="MaterialType"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="soils">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="0" name="soil" type="SoilType"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
```

Figure 10: CONCEPTS XSD file

XML Parser

The XML parser *CONCEPTSXMLParser* was created to extract information from an XML file to be inserted into the MYSQL database. The information that was obtained in the XML file is information about the environment being assessed and evaluated to create a successful stream restoration plan. Once the information is extracted, it is inserted into a database called CONCEPTS. The XML file has much database relational information, which is defined in the XSD (XML Schema Definition) file. The structure of each table relationship must be maintained so that information in the XML file is extracted and stored in a manner consistent with the XSD file.

The predefined classes used within the parser are the *Collections Dictionary* class and *MySQLConnection* class. The parser class also has user defined classes *XMLLoader, Document,* and *Element*. The *MySQLConnection* class was used to connect to the database. This class allows for many of the database functions such as SQL insert to be performed. The *Dictionary* class was used to maintain all the many-to-many and some of the one-to-many database table relationships. The user defined classes *XMLLoader, Document,* and *Element* were created to load and build the XML data. The *Element* class is where the majority of the data extraction takes place.

The *Element* class is used to represent each individual element (or tag) in the XML file. This class allows access to the data within the element. It also allows access to the parent and children of each element through the getParent, getChild, and getChildren methods defined in the *Element* class. Starting at the root element a depth first search exploration is used to traverse

each element listed under the root element and parse through the information. The relationships of the elements with each other are defined within the XSD file. Within the XSD file a many-to-many relationship among the elements is defined by the presence of maxOccurs = “unbounded”, ecore: reference = “-“, and type = “xsd:URI” in the tag attributes. See figure 11 below.

```
- <channelModels>
    - <channelModel name="Existing Conditions - All sections">
          <reach>#//@conceptsModel/@data/@reaches/@reach.0</reach>
      </channelModel>
    - <channelModel name="Existing Conditions - Coarse spacing">
          <reach>#//@conceptsModel/@data/@reaches/@reach.1</reach>
      </channelModel>
    - <channelModel name="New Alignment - All sections">
          <reach>#//@conceptsModel/@data/@reaches/@reach.2</reach>
      </channelModel>
    - <channelModel name="New Alignment - Coarse spacing">
          <reach>#//@conceptsModel/@data/@reaches/@reach.3</reach>
      </channelModel>
  </channelModels>
```

Figure 11: XML Example of Ecore Reference ID located in XML file

When these elements are encountered in relationships, they are represented by the ecore reference id which is a unique identifier within the XML file (shown highlighted in blue in Figure 11). These unique identifiers are nested within all of the elements and, within these elements have a relationship. Because the relationships in the XML file must be maintained in the database and each element’s information is only visited once but may be referenced several times, it then becomes necessary to save the element’s primary key after it is inserted in a table for later foreign key insertions.

```
<xsd:complexType name="ChannelModelType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      A channel model may comprise a single reach or a series of reaches
      separated by structures.  The first and last element of a channel
      model must be a reach.  The reaches and structures are references
      to their definitions in the data/reaches and data/structures
      elements.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element ecore:reference="ReachType" maxOccurs="unbounded" name="reach" type="xsd:anyURI"/>
    <xsd:element ecore:reference="CulvertSectionType" maxOccurs="unbounded" minOccurs="0" name="culvertSection" type="xsd:anyURI"/>
    <xsd:element ecore:reference="BridgeSectionType" maxOccurs="unbounded" minOccurs="0" name="bridgeSection" type="xsd:anyURI"/>
    <xsd:element ecore:reference="DropStructureType" maxOccurs="unbounded" minOccurs="0" name="dropStructure" type="xsd:anyURI"/>
    <xsd:element ecore:reference="GenericStructureType" maxOccurs="unbounded" minOccurs="0" name="genericStructure" type="xsd:anyURI"/>
    <xsd:element ecore:reference="TributaryType" maxOccurs="unbounded" minOccurs="0" name="tributary" type="xsd:anyURI"/>
    <xsd:element ecore:reference="LateralInflowType" maxOccurs="unbounded" minOccurs="0" name="lateralInflow" type="xsd:anyURI"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required">
    <xsd:annotation>
      <xsd:documentation>Unique name of channel model</xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
</xsd:complexType>
```

Figure 12: XSD with maxOccurs= unbounded for many-to-many relationship

Each element's ecore reference ID is mapped to the database table's primary key and maintained by a .NET Dictionary object. The .NET *Dictionary* class holds the collection of key/value pairs where the ecore reference ID is the key and the database primary key is the value. When an element's ecore reference ID is found in the XML file, the primary key is found and is easily inserted to indicate the relationship in a table. See figure 13 below for an element represented by its ecore reference ID in an XML file.

```
- <channelModels>
    - <channelModel name="Existing Conditions - All sections">
          <reach>#//@conceptsModel/@data/@reaches/@reach.0</reach>
      </channelModel>
    - <channelModel name="Existing Conditions - Coarse spacing">
          <reach>#//@conceptsModel/@data/@reaches/@reach.1</reach>
      </channelModel>
    - <channelModel name="New Alignment - All sections">
          <reach>#//@conceptsModel/@data/@reaches/@reach.2</reach>
      </channelModel>
    - <channelModel name="New Alignment - Coarse spacing">
          <reach>#//@conceptsModel/@data/@reaches/@reach.3</reach>
      </channelModel>
  </channelModels>
```

Figure 13: XML File Example of Many-to-Many ChannelModel_Reach

This string is used as the key value for the *Dictionary* key/value pair variables. The primary key that was generated when the element was inserted into the database is used as the value variable of the *Dictionary* key/value pair. For example the dictionary [xsid] is equal to a primary key that appears as dictionary [#//@conceptsmodel/@data/@reaches/@reaches.0] and is equal to primary key 97, this primary key is auto-generated by the database. Figures 14, 15, and 16 below show the function used to insert the information into a database, the database table definition for the channel model table, and the entities' relationship.

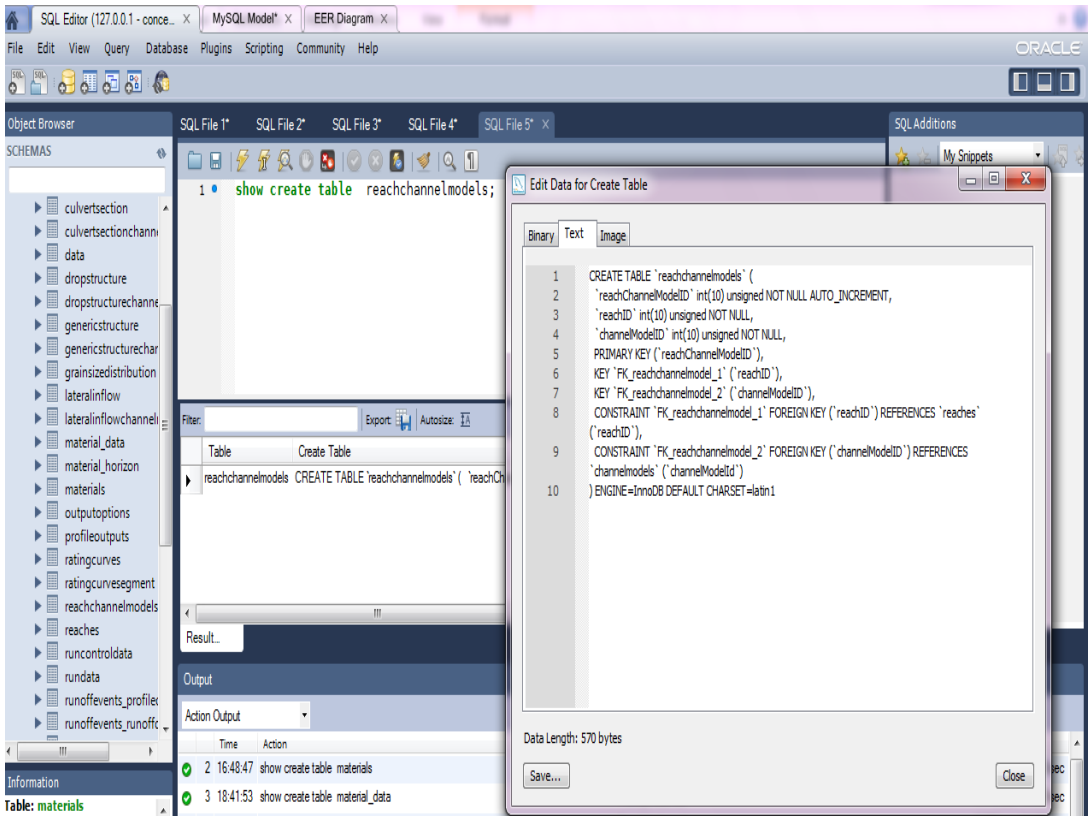


Figure 14: Reach-Channel Model Data Definition

```
// INSERT MANY TO MANY RELATIONSHIPS
// Reach-ChannelModel
foreach (Element reach in (List<Element>)channelModel.getChildren("reach"))
{
    String reachXSID = reach.getText();
    int reachID = xsidMap[reachXSID];
    query = "INSERT INTO reachchannelmodels (reachID, channelModelID) VALUES ("
            + reachID + ", " + channelModelID + ")";
    this.executeDBUpdate(query);
}
```

Figure 15: Implementation to store key/value pairs in dictionary class for Many-to-Many

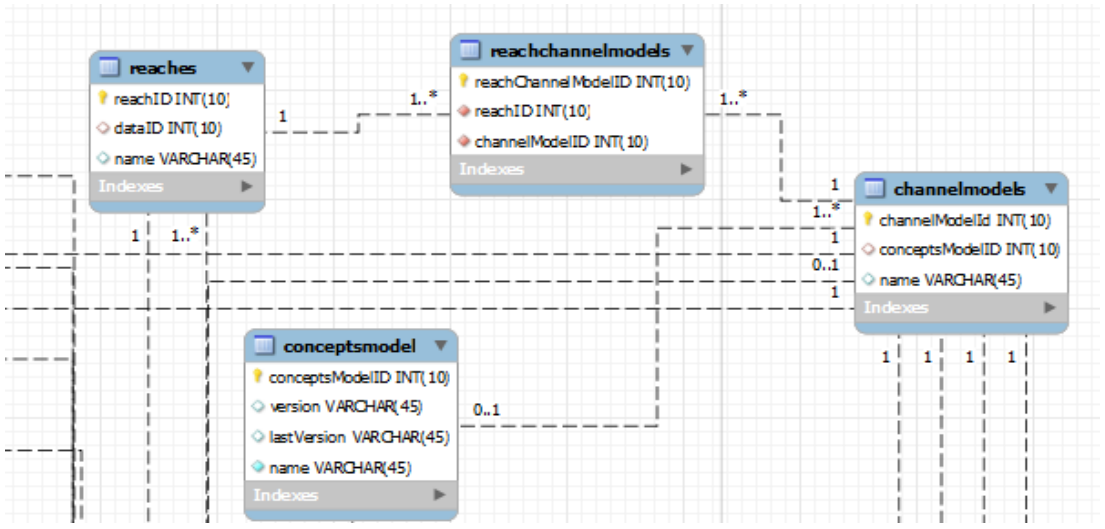


Figure 16: EER Diagram Showing the Reach-ChannelModel Relationship

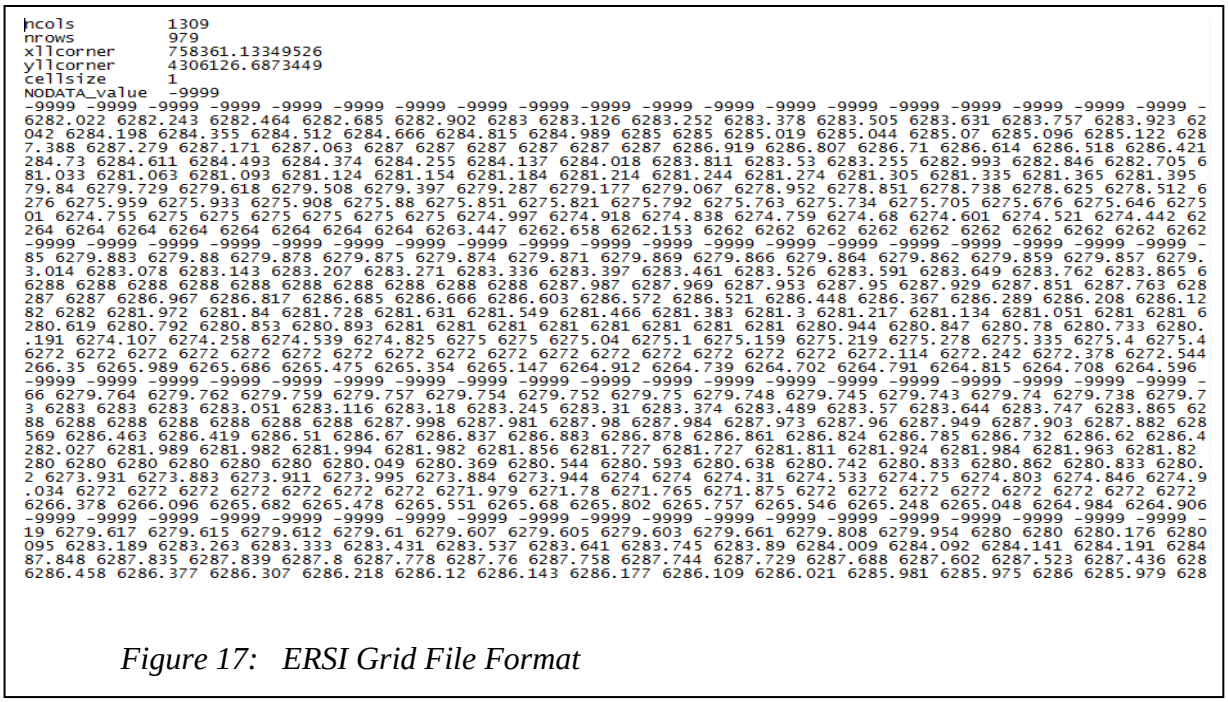
Storing the information using the Dictionary class allows for faster mapping and lookup operations of the XML file many-to-many and some one-to-many relational information. For all elements visited within the XML file they are extracted and stored into the database CONCEPTS. Again, the XML parser was only created to extract information about the terrain being investigated and the database was only created as a repository to house detailed information pertaining to the terrain.

Algorithm for the DEM files Parser

The second parser extracts data from the DEM file. The DEM file format is based on the ERSI file format and contains six fields in its header. The ncols field represents the number of columns the image or grid has. In the case of using an array the number of columns the 2D array would have. The nrows field represents the number of rows. The xllcorner and yllcorner are fields are the easting and northing coordinates of the lowest, leftmost point on the grid. The

cellsize indicates the length of one side a square cell of the grid. The NODATA_VALUE indicates a value that is given to cells with no known data. Figure 17 and 18 shows the DEM file format and a 2D array logical data structure representation.

The DEM file contains z (elevation) values. From the DEM file the x (easting) and y (northing) values are determined by the z value's position in the DEM file. The natural fit for this data is a 2D array. See figures 18 below. The values are used to produce a grayscale bitmap and raw HeightMap file from the elevation data. The idea is to use this information from the raw HeightMap to create a virtual reality terrain.



```
ncols         1309
nrows         979
xllcorner     758361.13349526
yllcorner     4306126.6873449
cellsize      1
NODATA_value  -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -
6282.022 6282.243 6282.464 6282.685 6282.902 6283 6283.126 6283.252 6283.378 6283.505 6283.631 6283.757 6283.923 62
042 6284.198 6284.355 6284.512 6284.666 6284.815 6284.989 6285 6285 6285.019 6285.044 6285.07 6285.096 6285.122 628
7.388 6287.279 6287.171 6287.063 6287 6287 6287 6287 6287 6287 6286.919 6286.807 6286.71 6286.614 6286.518 6286.421
284.73 6284.611 6284.493 6284.374 6284.255 6284.137 6284.018 6283.811 6283.53 6283.255 6282.993 6282.846 6282.705 6
81.033 6281.063 6281.093 6281.124 6281.154 6281.184 6281.214 6281.244 6281.274 6281.305 6281.335 6281.365 6281.395
79.84 6279.729 6279.618 6279.508 6279.397 6279.287 6279.177 6279.067 6278.952 6278.851 6278.738 6278.625 6278.512 6
276 6275.959 6275.933 6275.908 6275.88 6275.851 6275.821 6275.792 6275.763 6275.734 6275.705 6275.676 6275.646 6275
01 6274.755 6275 6275 6275 6275 6275 6275 6274.997 6274.918 6274.838 6274.759 6274.68 6274.601 6274.521 6274.442 62
264 6264 6264 6264 6264 6264 6264 6264 6263.447 6262.658 6262.153 6262 6262 6262 6262 6262 6262 6262 6262 6262 6262
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -
85 6279.883 6279.88 6279.878 6279.875 6279.874 6279.871 6279.869 6279.866 6279.864 6279.862 6279.859 6279.857 6279.
3.014 6283.078 6283.143 6283.207 6283.271 6283.336 6283.397 6283.461 6283.526 6283.591 6283.649 6283.762 6283.865 6
6288 6288 6288 6288 6288 6288 6288 6288 6288 6288 6287.987 6287.969 6287.953 6287.95 6287.929 6287.851 6287.763 628
287 6287 6286.967 6286.817 6286.685 6286.666 6286.603 6286.572 6286.521 6286.448 6286.367 6286.289 6286.208 6286.12
82 6282 6281.972 6281.84 6281.728 6281.631 6281.549 6281.466 6281.383 6281.3 6281.217 6281.134 6281.051 6281 6281 6
280.619 6280.792 6280.853 6280.893 6281 6281 6281 6281 6281 6281 6281 6281 6280.944 6280.847 6280.78 6280.733 6280.
.191 6274.107 6274.258 6274.539 6274.825 6275 6275 6275.04 6275.1 6275.159 6275.219 6275.278 6275.335 6275.4 6275.4
6272 6272 6272 6272 6272 6272 6272 6272 6272 6272 6272 6272 6272 6272 6272 6272 6272 6272 6272 6272.114 6272.242 6272.378 6272.544
266.35 6265.989 6265.686 6265.475 6265.354 6265.147 6264.912 6264.739 6264.702 6264.791 6264.815 6264.708 6264.596
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -
66 6279.764 6279.762 6279.759 6279.757 6279.754 6279.752 6279.75 6279.748 6279.745 6279.743 6279.74 6279.738 6279.7
3 6283 6283 6283 6283.051 6283.116 6283.18 6283.245 6283.31 6283.374 6283.489 6283.57 6283.644 6283.747 6283.865 62
88 6288 6288 6288 6288 6288 6288 6287.998 6287.981 6287.98 6287.984 6287.973 6287.96 6287.949 6287.903 6287.882 628
569 6286.463 6286.419 6286.51 6286.67 6286.837 6286.883 6286.878 6286.861 6286.824 6286.785 6286.732 6286.62 6286.4
282.027 6281.989 6281.982 6281.994 6281.982 6281.856 6281.727 6281.727 6281.811 6281.924 6281.984 6281.963 6281.82
280 6280 6280 6280 6280 6280 6280.049 6280.369 6280.544 6280.593 6280.638 6280.742 6280.833 6280.862 6280.833 6280.
2 6273.931 6273.883 6273.911 6273.995 6273.884 6273.944 6274 6274 6274.31 6274.533 6274.75 6274.803 6274.846 6274.9
.034 6272 6272 6272 6272 6272 6272 6272 6272 6271.979 6271.78 6271.765 6271.875 6272 6272 6272 6272 6272 6272 6272
6266.378 6266.096 6265.682 6265.478 6265.551 6265.68 6265.802 6265.757 6265.546 6265.248 6265.048 6264.984 6264.906
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -
19 6279.617 6279.615 6279.612 6279.61 6279.607 6279.605 6279.603 6279.661 6279.808 6279.954 6280 6280 6280.176 6280
095 6283.189 6283.263 6283.333 6283.431 6283.537 6283.641 6283.745 6283.89 6284.009 6284.092 6284.141 6284.191 6284
87.848 6287.835 6287.839 6287.8 6287.778 6287.76 6287.758 6287.744 6287.729 6287.688 6287.602 6287.523 6287.436 628
6286.458 6286.377 6286.307 6286.218 6286.12 6286.143 6286.177 6286.109 6286.021 6285.981 6285.975 6286 6285.979 628
```

Figure 17: ERSI Grid File Format

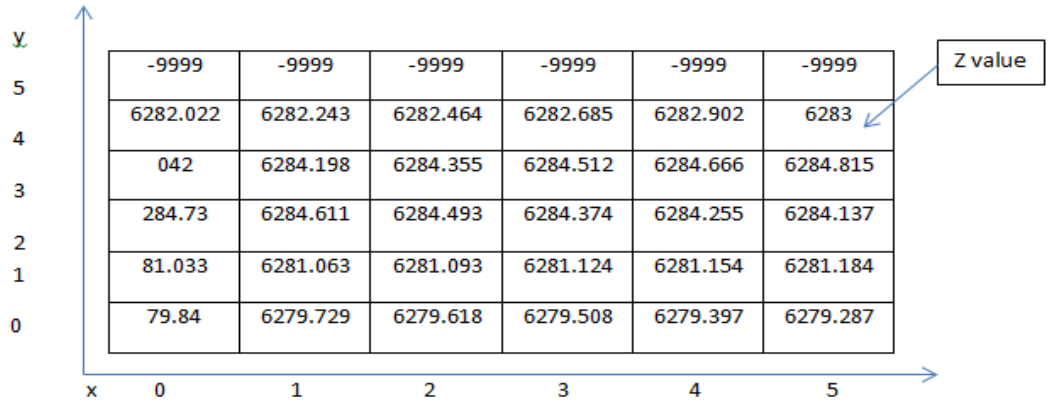


Figure 18: 2D Array of DEM Grid File

The parser is implemented by first reading the ASCII (DEM) file header and extracting its field values. After this, the parser searches each individual line and separates the values on the lines based on the space delimiter. The native coordinate system of the DEM file is not preserved. The native coordinate system is not applicable because the coordinate system for the new files created starts in the upper left hand corner and goes rightward and downward. The values in the DEM file are read in the same manner. The values are then placed into a 2D array used for creating a grayscale bitmap for display. After the bitmap is displayed a 256 X 256 pixel area is selected from the image. Each pixel represents 1 square meter. The 256 X 256 pixel area becomes a data subset from which a Raw32 file is created for the gaming engine.

Algorithm for Creating a BitMap from the DEM File

For creating a bitmap, the module starts by retrieve the image range, by taking the lowest and the highest elevation read. Then dividing the image range by 256 provides the scale factor

for the objects in the in the virtual environment. When processing each value in the 2D array, each elevation value has lowest elevation in the entire dataset subtracted from it to calculate a range starting at 0. Once the lowest elevation is subtracted each value is then multiplied by the scale factor. This range value ensures that the value falls in the range of 0-255. This value provides the best contrast for a grayscale bitmap. If the values are within a small range it is harder to distinguish the different elevations within the bitmap. For example given the smaller range 0-20 when all values are close together the contrast between colors is very close where distinction could not be made, but when scaled to 0-255 a greater variation of the black and white colors are applied. Therefore, using greater variation, distinction can be better determined because the low values are darker and the high values appear lighter. See figure 19.

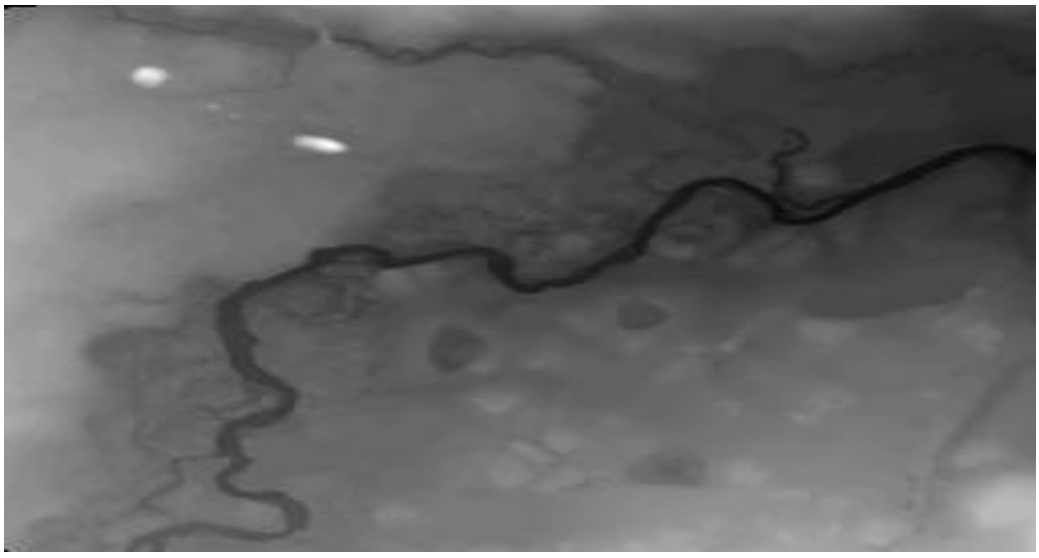


Figure 19: Resulting BitMap File of DEM Data Using Algorithm Above

Each value calculated is used for pixel color values in the bitmap. The code starts with the first cell in the 2D array and applies a grayscale color value to a pixel based on the 0-255

range value. This scaling operation is done using the equation below.

pixel value for all the channel = (value in 2d array – lowest elevation) * scale;

The bitmap is initialized with the same dimensions as the array. The coordinate system is the same where the top leftmost pixel is (0,0) and the x values increase to the right and the y values increase going down. When there is no data value assigned in a cell the color appears black. Once processed the bitmap image appears in a window pane called the Image Viewer. See figure 20. At the current time the image viewer renders as a separate window when the CONCEPTS3D desktop application is executed.

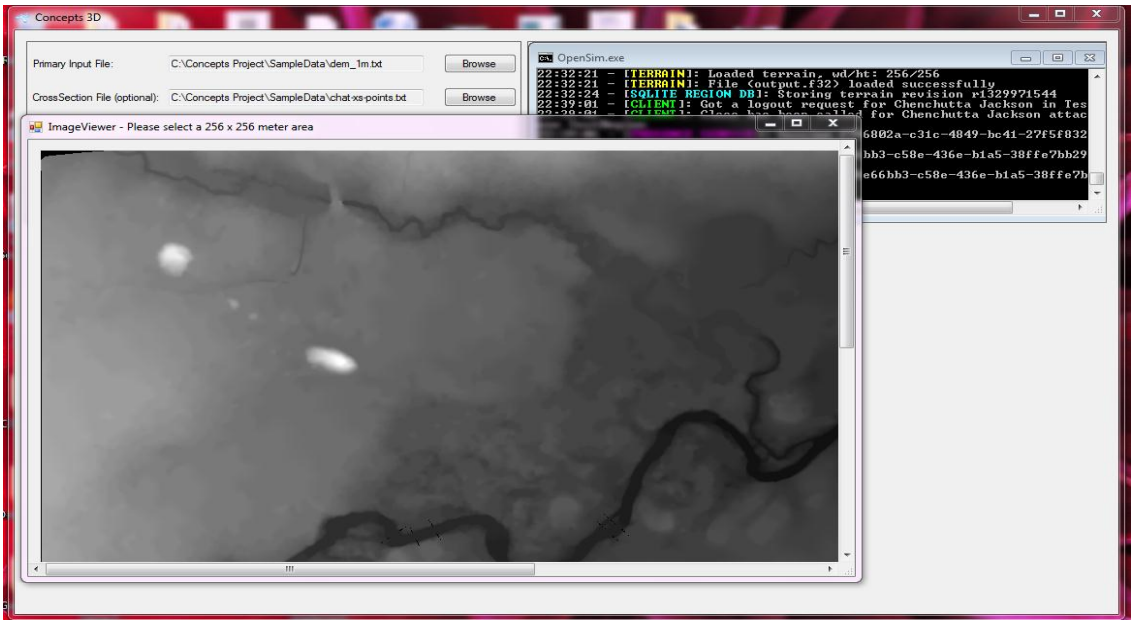


Figure 20: Grayscale BitMap of the Integrated XSection and 1 Meter Dem File

Once the image is in the Image Viewer, the user can click the mouse in the panel. A 256 x 256 rectangle box will be drawn around a specified area from here, and then a RAW32 image is created to be used by the gaming engine. See figure 21 below.

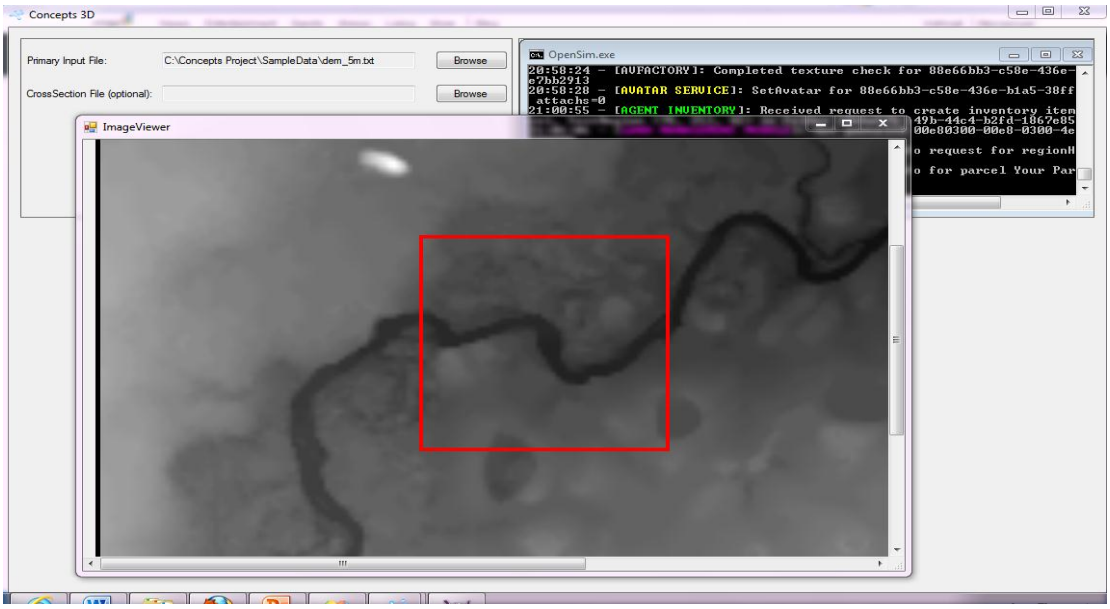


Figure 21: Selected 256 X 256 Area of the 5 Meter Dem File

Algorithm for Creating the X-Section Parser

The X-Section parser was implemented much like the DEM parser. However, here the file format already had x, y, and z, coordinates embedded representing easting, northing, and elevation respectfully. (1) The X-Section parser component starts by reading the file and extracting each value in the line of the comma separated file. (2) Once this data is extracted, it is stored in a list of a user defined Point3D objects. The *Point3D* class has x (easting), y (northing), and z (elevation properties for storing data). (3) This data is merged with the DEM file data by looping through the POINT3D list merging the PointList with the DEM information. The x and y coordinates tell us where in the 2D array to insert the data value. However, because the gaming engine's coordinate system cannot go lower than a square meter and the X-Section file have easting and northing values in fractions of meters there is a significant loss in precision. The

decimal portion of the easting and northing values is truncated. This loss of precision ultimately causes the X-Section data to overwrite existing DEM file data. (This loss of data precision does not affect the visualized object.) The z value is scaled down in a similar way to the DEM file parser. Once this elevation value is calculated, it is placed directly in the array, based on the x and y value (i.e. elevationData[x,y] = elevationValue).

Algorithm for Creating the Raw32 File

The RAW32 file is created from a subset of the 2D array of the DEM file data and is based on the location of the 256 X 256 area selected within the bitmap (See Figure 22).

Figure 22: BitMap 256 X 256 Area Selected

The RAW32 file module loops through the subset of the 2D array. Each value is written to the RAW32 file as a single precision floating point binary number.

The proof of concept to visualize CONCEPTS 2D data as a 3D terrain in a virtual world began by trying to open the bmp files created by the heightmap generator see figure 7 on page 60 in the game engine, NeoAxis. There was no success with the NeoAxis gaming engine because this particular gaming engine requires an unorthodox file format to render a terrain. This effort made clear the fact that a standard file format had to be designed and used throughout the rest of this project. So another gaming engine was chosen, which was also based on the .NET framework called OpenSim.

This gaming engine provided a way for the proof of concept to be validated with a variety of common image formats. The first format tried was the bmp format because of its ease of use; this format worked. Terrain size appeared to be limited to 256X256 pixels. Looking at sample data of OpenSim terrain files showed that with the RAW32 file format, the user was allowed to load larger terrains and this was desirable.

Inspection of source code made it clear that the RAW32 format was the native format used by the gaming engine to represent terrain height values. Data values in the other file formats such as bmp, gif, png, and tiff needed some form of conversion before being used by OpenSim. With the RAW32 format, the floating point values are read directly from a byte stream. Each value in the RAW32 file is represented by an IEEE 32-bit single precision floating point number.

Before the above process was completed the first task performed was creating the database that was used to maintain CONCEPTS and CONCEPT3D data. The database management system used for this project was MySQL. The existing CONCEPTS XSD was used

to create the relational schema, table formats, and queries.

How CONCEPTS should be executed was the next decision that was made. CONCEPTS can be executed in two ways. CONCEPTS can run as a standalone model, which requires the user to download CONCEPTS on his or her machine. The second technique was to run CONCEPTS via the web. The results from the CONCEPTS simulator are saved in the database. Currently, the results produced from the CONCEPTS simulator generate a text file that is saved in an SQL database. Currently, the output of the simulator is imported into a spreadsheet application. From the spreadsheet application the user can view the results in plot or graph form (See Figure 23).

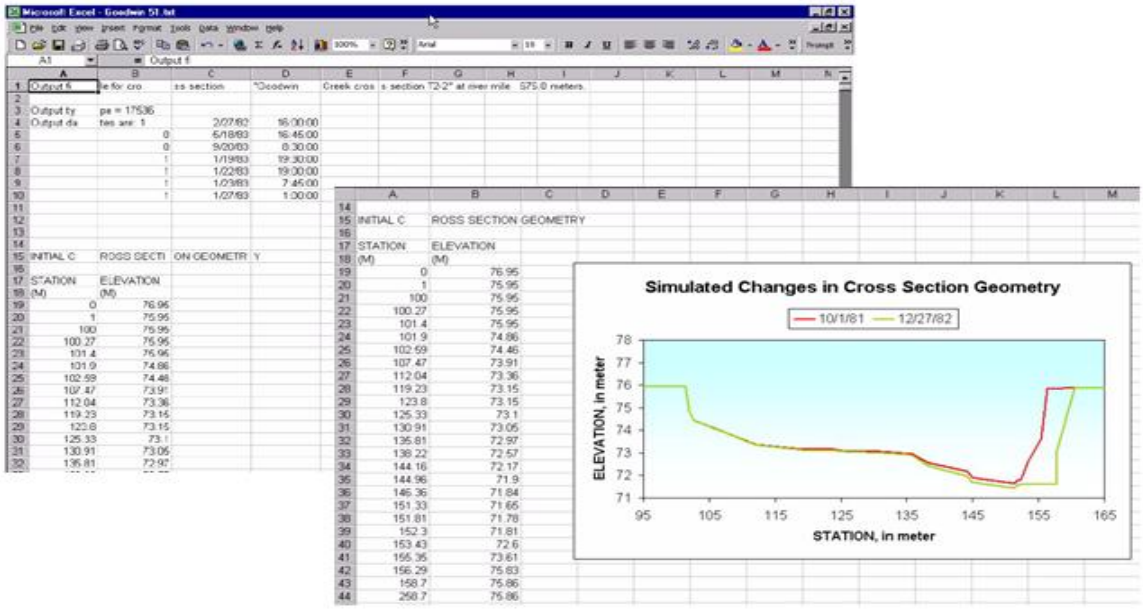


Figure 23: CONCEPTS Data Display in Excel

For testing CONCEPTS3D, executing CONCEPTS as a standalone desktop application was not required. Much of the data for testing CONCEPTS3D was provided by the researchers at

the Sedimentation Laboratory. The data provided was an XML file, DEM 1 and 5 meter text file, and an EXCEL X-Section file.

Data Enhancement

Before rendering the 3D terrain for CONCEPTS 2D 5 meter dataset, the information or data supplied to the gaming engine must be modified and enhanced using bilinear interpolation and Delaunay triangulation. The first file modified was the CONCEPTS DEM file. This file only supplies the z data value for the 3D model to be rendered. The x and y value must be determined by the location of the z value within the DEM file, see figure 18, page 72. With the x, y, and z values obtained, a 3D model can be created from the available information. From here, data enhancement techniques are performed by interpolation and triangulation methods to create a continuous dataset from the discrete dataset. Finally, the dataset points and triangles are reduced to optimize computing efficiency and visualization refinement. This function is built-in to the gaming engine OpenSim and only had to be accessed with correct data formats. The code for each module created can be found in the appendix section on page 118 of this dissertation.

The other files that underwent similar data modifications and enhancement processes are CONCEPTS's X-Section files. So the data supplied to the gaming engine was provided as two separate files. One file is information about the terrain floodplain (DEM) and the other file is about the channel X-Section that lies within the floodplain. These two files were merged to create one complete file about the topography of the terrain under investigation. The creation of this larger dataset required an additional data processing mechanism before being rendered by the gaming engine. See figure 30 (small 5 meter data) and 33 (larger 5 meter data) on page 91 and 93.

Enhancing the data set to achieve and visualize a better rendered image is one of the most valuable and important components. Enhancing the dataset with interpolation and triangulation produced a more realistic image based on the information provided from CONCEPTS simulation model and the other various files. The interpolation techniques used were bilinear interpolation, which is an extension of linear interpolation and Delaunay triangulation with Barycentric interpolation. Each of the module functions created to enhance the dataset is described in detail below.

Linear Interpolation/Bilinear Interpolation

Linear interpolation is the easiest interpolation method to grasp and implement. Linear interpolation is a method of approximating the graph of a function in between two points on the graph by the straight line between them. Linear interpolation is defined as given two known points of the coordinate system (x0, y0) and (x1, y1), the linear interpolation is the straight line between these two points. To calculate the y value from existing values to create a continuous data set the equation is provided below.

> Linear interpolation involves estimating a new value by connecting two adjacent known values with a straight line. If the two known values are (x0, y0) and (x1, y1), then the y value for some point x is:
>
> $$y = y_1 + (x - x_1)\frac{y_2 - y_1}{x_2 - x_1}$$ (Blue Leaf Software, 2010)

The interpolation method used for this dissertation is bilinear. Bilinear interpolation is an extension of linear interpolation meaning that instead of interpolation done in one direction, interpolation is done in both the x and y direction . Bilinear interpolation produces smooth and

continuous data, which produces a realistic dataset for rendering in 3D visualization. Bilinear interpolation performs interpolation by using four points known and finds the weighted approximate value of the unknown points to provide additional points for the discrete data set.

Delaunay Triangulation with Barycentric Interpolation

Delaunay triangulation with barycentric interpolation was the other method used to enhance the discrete data set provided by CONCEPTS. Delaunay triangulation is defined as a n-dimensional triangulation of a set of points P = (p1,p2,p3,….pN). In other words, Delaunay triangulation is a collection of n-dimensional simplices whose defining points lie in P. Simplices are the convex region defined by a set of n+1 independent points such as a 1D-point, 2D-line, 3D-triangle, etc. The simplices do not intersect one another and share only boundary features such as edges or faces. Delaunay triangulation has the property that the circumsphere of any n-dimensional simplex contains no other points of P except the n+1 defining points of the simplex. The techniques used to compute the Delaunay triangulation of this dissertation were defined by Watson and Bowyer.

The algorithms begin by constructing an initial Delaunay triangulation that strictly bounds the points of P see figure 24 below. Then each point of P is inserted or injected one by one into the current triangulation. If the inserted point lies within the circumcircle of any simplex, then the simplex was deleted creating a hole in the triangulation. Once simplices were deleted the n-1 dimensional faces of the boundary of the hole along with the inserted point were used to construct a modified triangulation. This process continues until all the points were inserted into the triangulation. The final step was to remove the simplices connecting the points forming the initial bounding triangulation creating the Delaunay triangulation [Schroeder, Marin,

& Lorensen, 2004].

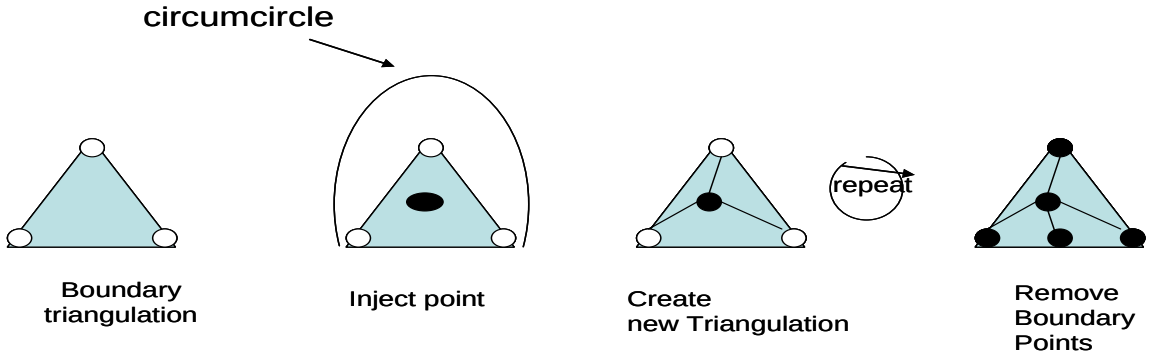


Figure 24: Delaunay Triangulation Algorithm [Schroeder, Martin, and Lorensen, 2004]

To calculate more data points for an enhanced dataset, interpolation was performed. The interpolation method used after Delaunay triangulation was performed is barycentric interpolation. The general definition of barycentric interpolation or barycentric coordinates is that it is an interpolation method that uses the three defined points that make up a triangle to calculate the unknown points within the triangle. A detail definition of barycentric coordinates is stated below.

Given a 2D triangle with vertices p0, p1, p2 € R. Let x € R be any point in the plane.
The unknown x value is calculated by: $x = \alpha p0 + \beta p1 + \gamma p2$ where $\alpha + \beta + \gamma = 1$.
The point lies within the plane if and only if $\alpha, \beta, \gamma \in [0,1]$ [McAllister, 2007]. The point coordinate is calculated based on it relativity to the three points defined within the plane.

After these functions were implemented a method was developed to reduce the amount of data produced by the interpolation and triangulation methods. The goal was to use a level of detail technique to reduce the amount of data needed to render an image and to optimize the

entire rendering and visualization process, this mechanism is provided by the gaming engine. Each function for enhancing the dataset is provided in the appendix section page 122 of this dissertation.

Testing

For the purpose of this prototype development project, all testing was focused on "Did it work", not, "How effective was the resulting product". Did the application provide a way to visualize CONCEPTS terrain data as a 3D virtual reality environment? The prototype of this dissertation worked as expected. The user of this application can view CONCEPTS data as a 3D virtual environment by using the OpenSim's gaming engine. The question of the effectiveness of this application is outside the scope of this dissertation and can be evaluated for further knowledge.

In the testing phase of the SDLC, the logical internal software is reviewed and evaluated. Each module created is thoroughly evaluated and tested using unit testing. After each individual module is combined, then the aggregated system is tested as one seamless and complete model. For this project the component-CONCEPTS3D desktop application with each parser unit and the gaming engine OpenSim were integrated and tested. The CONCEPTS model runs as a desktop application. The datasets used for testing the system was the golf course area of Lake Tahoe in California (including the 1 meter and 5 meter DEM file, the combined X-Section and 1 meter DEM file, and GC_analysis CONCEPTS XML file). The results of each file are discussed in the Results section of this dissertation.

The gaming engine used for testing was NeoAxis and OpenSimulator (OpenSim). Because NeoAxis used an unorthodox file format, creating test cases for NeoAxis were not

achievable or obtainable. Eventually, the OpenSim game engine was the engine used for testing all test cases. OpenSim is a multi-user and multi-platform 3D web application server ["OpenSim"]. The gaming engine accepted many of the files created to be tested with ease. The primary reason OpenSim was used because it was easily extensible because of the basic approach taken when the application was developed. Like Neoaxis, OpenSim is written in C# and it also has the capability to run on Windows and Unix-like machines through the use of the .NET framework and Mono Runtime framework. The better feature of OpenSim is that it is truly an open source engine provided by Berkeley Standard Development (BSD). This license allows developers to create seamless application by providing an easy way to integrate other modules that plug-in to the applications with source code at their disposal. With using OpenSim, many implementation details about how the data is processed before and after rendering takes place is provided. This was helpful because the file created for the virtual reality terrain was a customized environment. The file format used by the gaming engine was an important factor to the success of rendering the customized data. Also scaling an object within the virtual environment was important information to know. With the engine being open source many cumbersome implementation details are available for developers to modify and extend as needed.

Maintenance

The final step in the SDLC is the maintenance phase. Software inevitably will need to be modified either because of system architecture changes or just code redesign. The reasons software undergoes changes are almost innumerable. Therefore, software must be developed in a manner that will allow modification effortlessly. Because the software developed for this

dissertation was a prototype, maintenance is not a critical issue at this time.

Limitations of CONCEPTS3D

Currently the limitations of CONCEPTS3D are as follows:

1. CONCEPTS3D is not accessible remotely; it is only available as a standalone desktop application.
2. CONCEPTS3D is only available as a Windows application.
3. The terrain rendered by CONCEPTS3D is limited to 256 meters x 256 meters.
4. No history of previous terrains rendered is maintained by the application.

CHAPTER IV

RESULTS

The primary objective of this dissertation was to determine if graphically visualizing in a 3D gaming environment using the data provided from CONCEPTS 2D numerical model was feasible. One goal was to take the CONCEPTS data, modify and enhance it to produce realistic models of the terrain under investigation, all the while maintaining the integrity of the information provided from the CONCEPTS numerical model. To test CONCEPTS3D several files were tested: 1 meter DEM data, 5 meter DEM data, the X-Section combined with 1 meter DEM data, and the CONCEPTS XML file. The data enhancement techniques, bilinear interpolation and Delaunay triangulation with Barycentric interpolation, prove to provide much needed data points for the 5 meter DEM sample data used to render a terrain. The results received by the data enhancement techniques were required to render the terrain by the gaming engine. For the 1 meter sample data provided, these techniques proved to be excessive. The 1 meter data was the highest precision that can be used by the gaming engine to render a model effectively. In this case there was no need to enhance the data because the one 1 meter data provided adequate granularity.

Before results can be viewed, several steps were required to prepare the data for rendering by the gaming engine.

1. Start the gaming engine OpenSim (Figure 25, page 87).

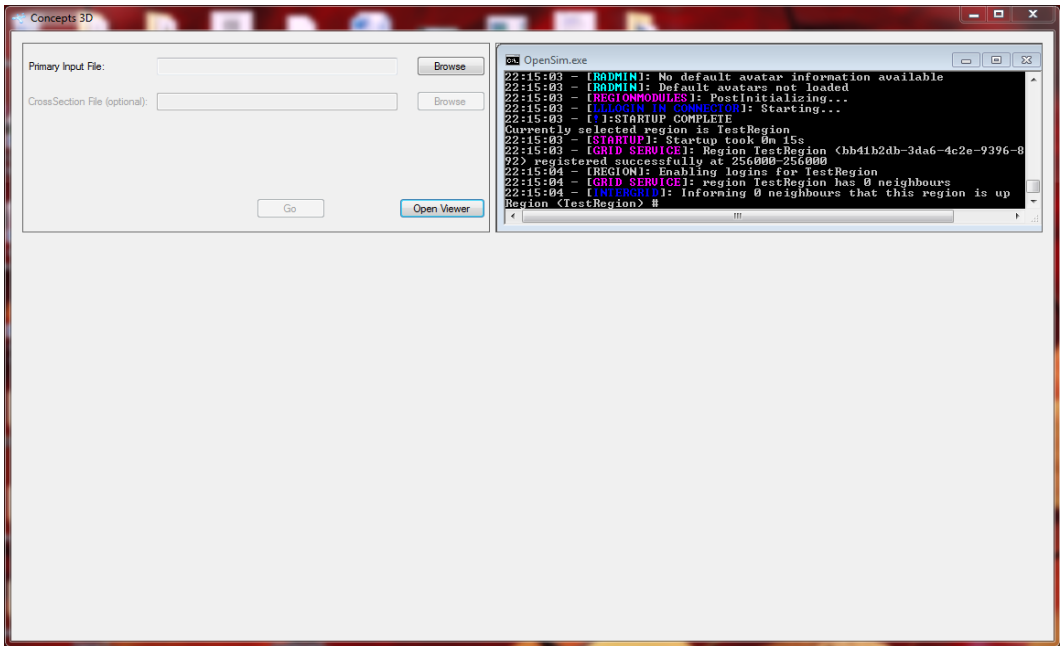


Figure 25: Start of the OpenSim Game Engine

2. Choose a file for rendering (Figures 26 and 27, page 88).

3. Log into the image viewer located in the bottom portion of the CONCEPTS3D application window pane (Figure 8, page 62).

4. The user is then asked to select an area from the file for viewing.

Each step is shown in the figures below.

1. *The user clicks the start button in the right upper portion of the CONCEPTS3D desktop application to start the OpenSim gaming engine.*

2. *Step 2 the user selects a file or files for data processing. Here the user chooses to process the 1 meter DEM file and X-Section file. The files are displayed in the left portion of the CONCEPTS3D desktop application graphical user interface (GUI). The input box is filled with the name of the files being processed. The user then clicks "Go" to start data processing. (See figures 26 and 27).*

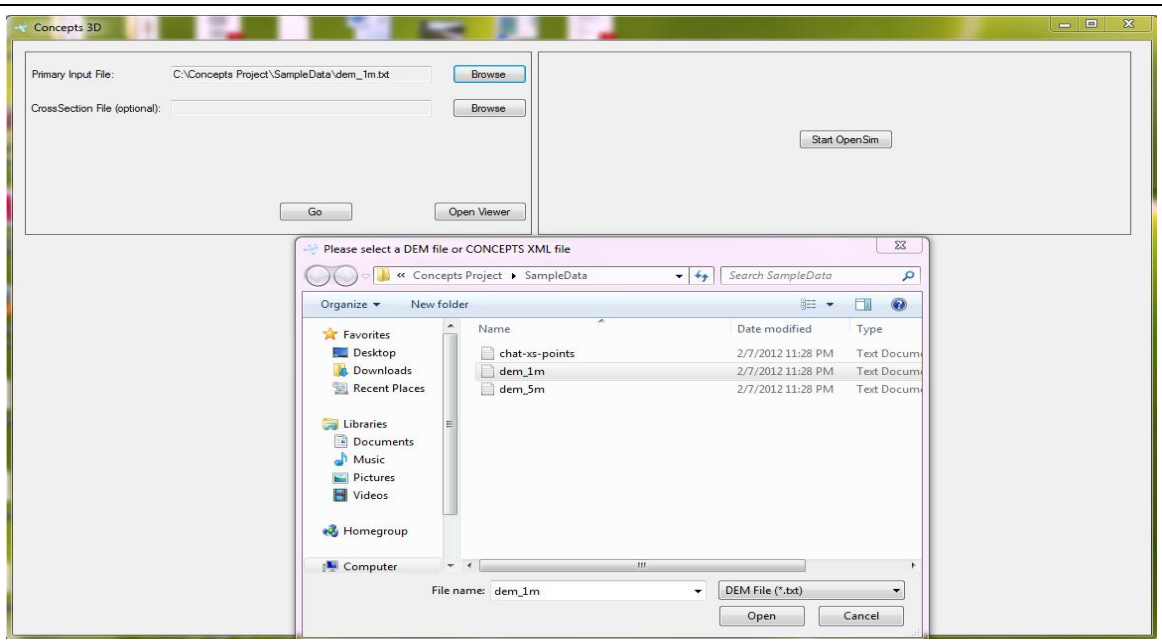


Figure 26: Choose a DEM file for Rendering

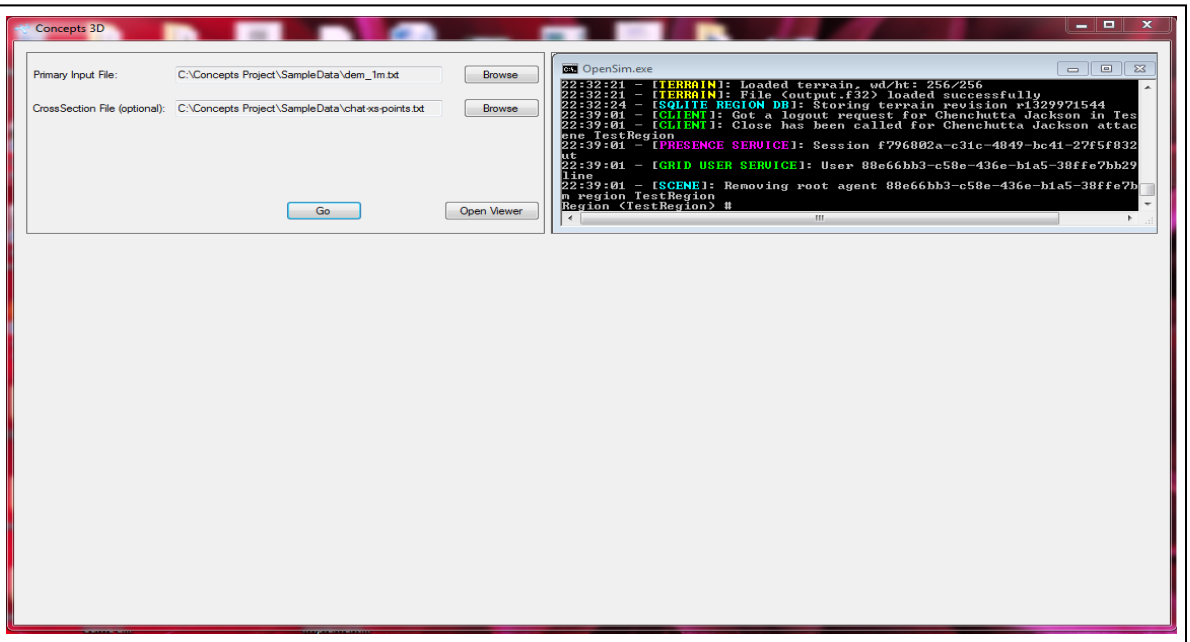


Figure 27: Choose files or a file to render

3. *Step 3 the user presses the image viewer button in the bottom portion of the*

CONCEPT3D desktop application. The image viewer allows the user to view the data being processed as a 3D virtual terrain.

4. *The OpenSim displays the virtual environment in the image viewer as a 3D virtual environment.*

Figure 28: View the Selected area as a 3D virtual environment

The results above figure 28 reflect the outcomes of processing the 1 meter DEM data file supplied to the gaming engine before bilinear interpolation and Delaunay triangulation are performed to the dataset along with the image/model produced from it.

The next figures show the result from testing the 5 meter data set.

1. *The user selects the 5 meter DEM file for processing (See Figure 29).*

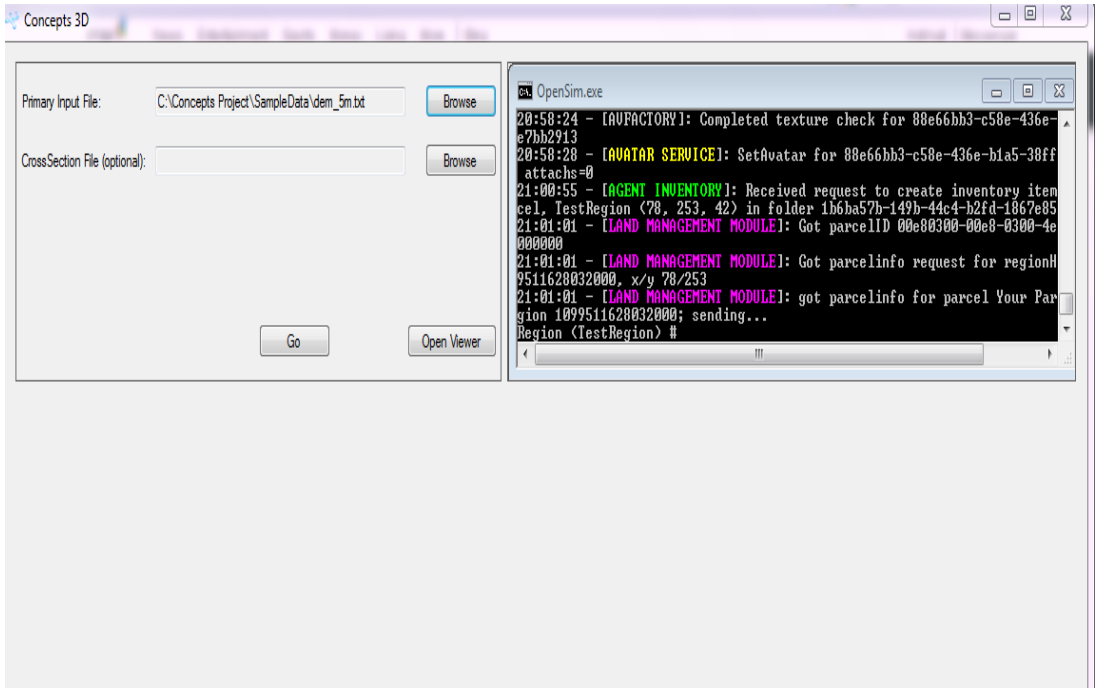


Figure 29: User Selecting a 5 meter Dem file for Viewing

2. *The user then clicks "Go" to start data processing. (See Figure 30).*

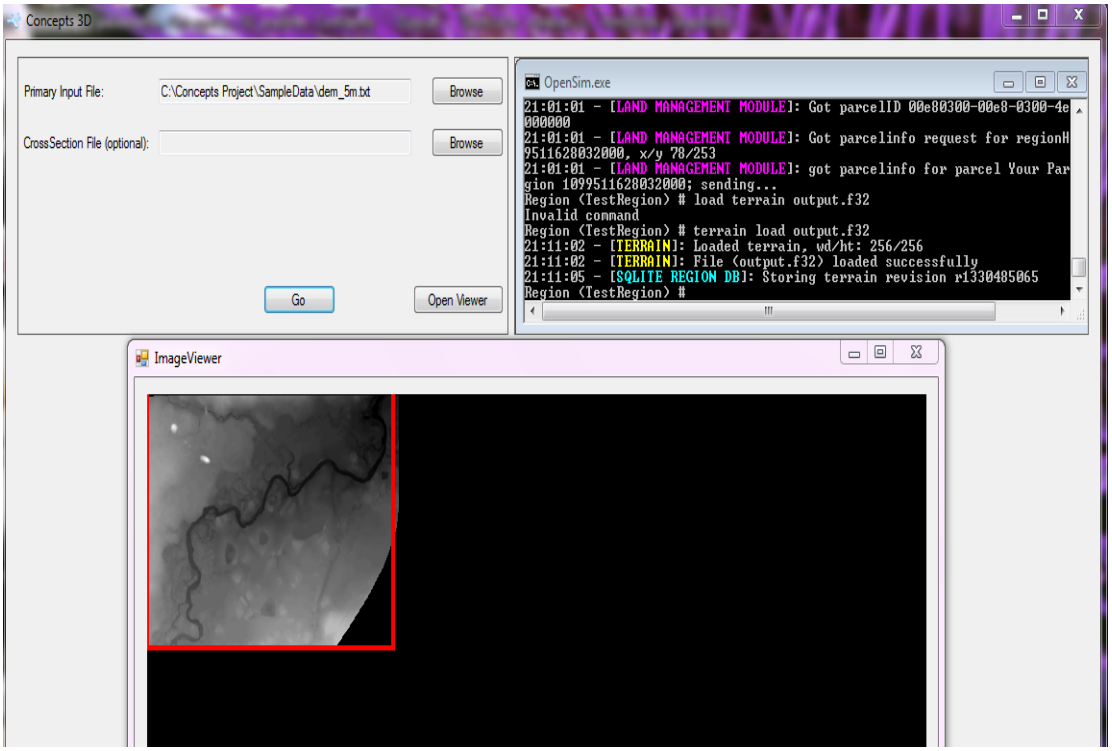


Figure 30: The Results of a 5 meter Dem file as a BitMap

3. *Step 3 the user presses the image viewer button in the bottom portion of the CONCEPT3D desktop application menu selection pane (See Figure 8 and 30). The image viewer allows the user to view the data being processed as a 3D virtual terrain.*

4. *The OpenSim displays the virtual environment in the image viewer as a 3D virtual environment.*

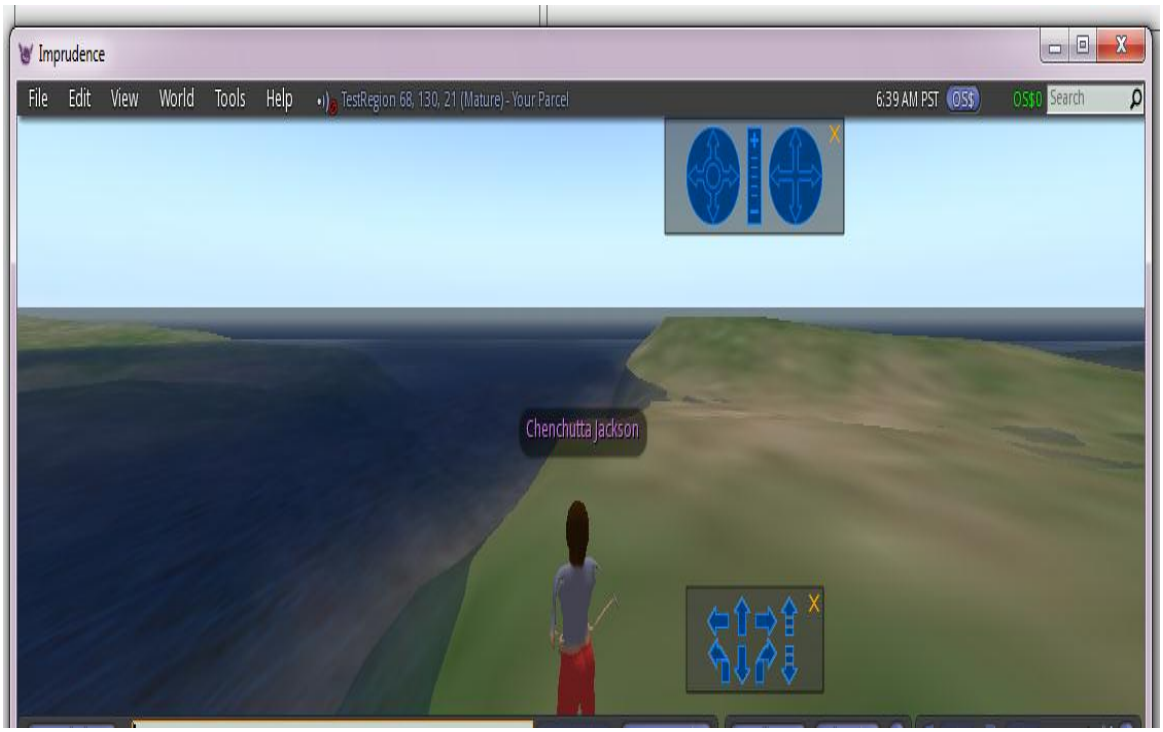


Figure 31: Results of Dem 5 Meter Data as a VR Terrain After Interpolation

The result of the parsed 5 meter DEM file without further interpolation is not sufficient to render a virtual terrain see figure 30 above. This is due to the lack of data points within the dataset to render a 3D model. The number of pixel values that must exist to render the virtual terrain is minimally 256 x 256. Consequently, before the results could be rendered by the gaming engine, data enhancement was required for the 5 meter data set. The data enhancement technique used was bilinear interpolation and Delauney triangulation with Barycentric interpolation. The result of each data enhancement technique performed on the rendered image is shown below.

The user is required to choose the file to parse and visualize as a 3D virtual environment. Once the file is chosen, the user selects the Go button, and then the HeightMap is displayed in the image viewer pane.

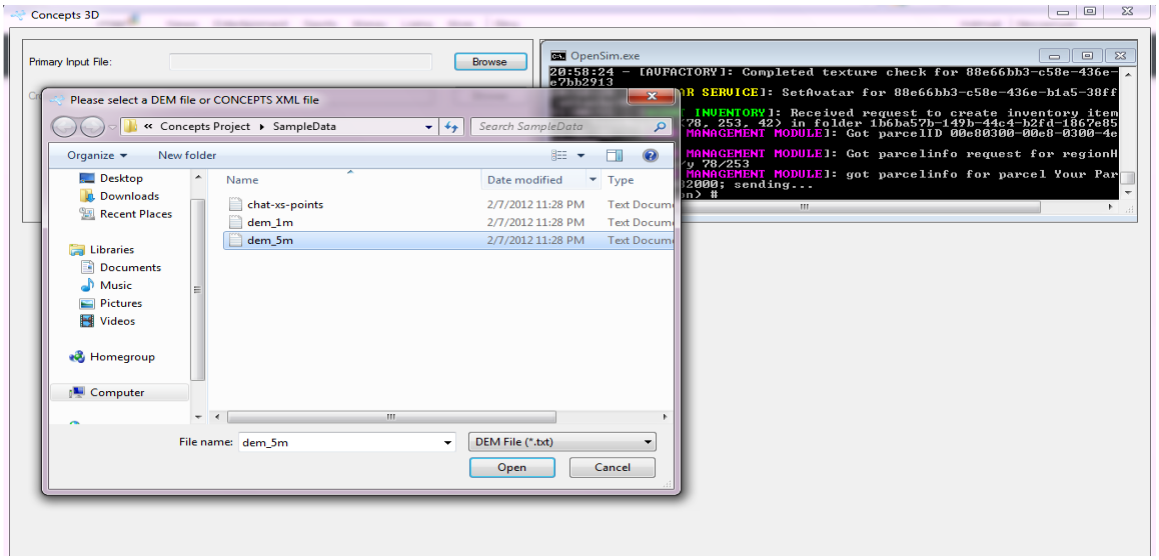


Figure 32: User Selecting 5 Meter Dem File

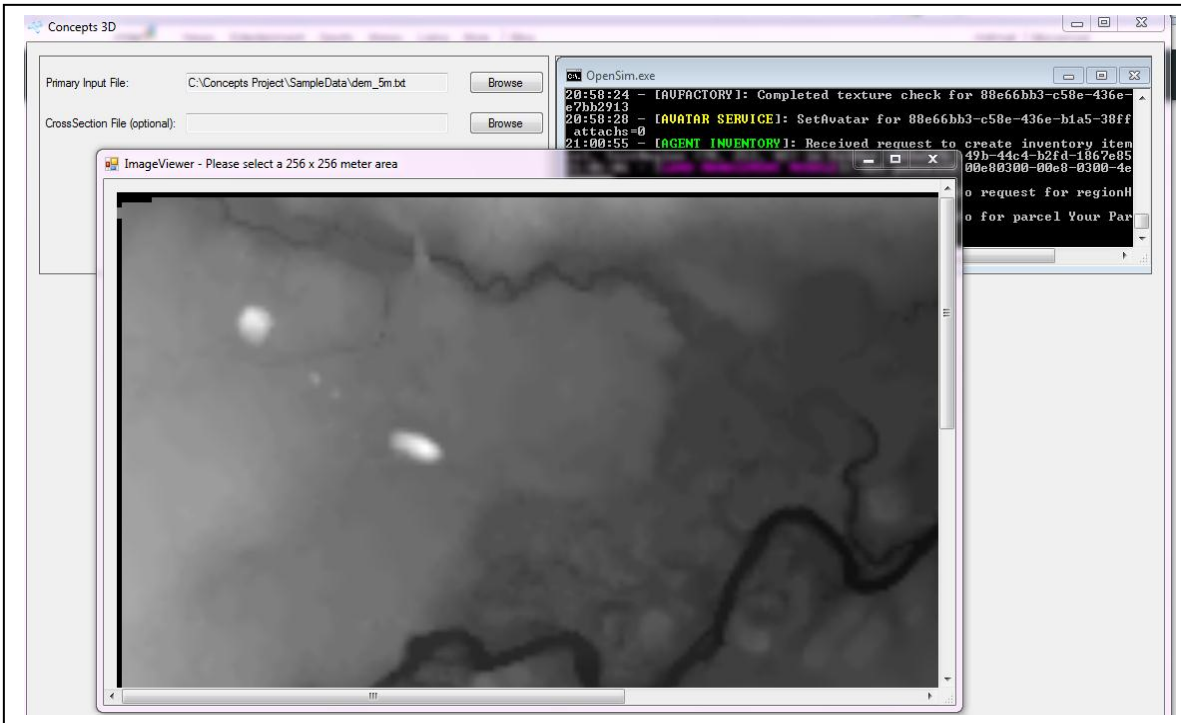


Figure 33: Results of 5 Meter Dem file after Bilinear Interpolation

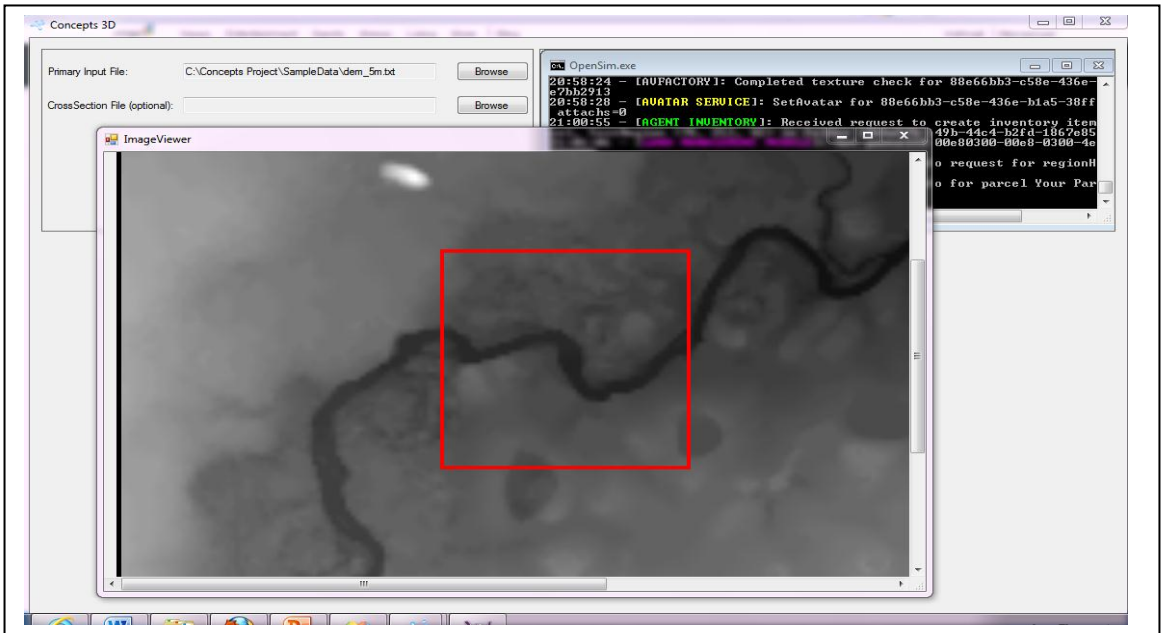


Figure 34: A 256 X 256 Selected Area for rendering an object from the 5 Meter Dem

As the figure 33 shows, for bilinear interpolation creating additional points allowed the image to be rendered by the gaming engine. OpenSim requires data to be 256 X 256, without data enhancement the data or bitmap created is 256 X 197 (See Figure 30). So after data enhancement, the bitmap was created to be of a standard size. Selecting a smaller area of 256 X 256 allows the image to be rendered by the gaming engine (See Figure 34). In the bitmap where data does not exist the image appears jagged. This is because the game engine tries to interpolate points that do not exist. In the virtual world where the data was interpolated and no data exists, huge spikes in the terrain appear (See Figure 35 below).

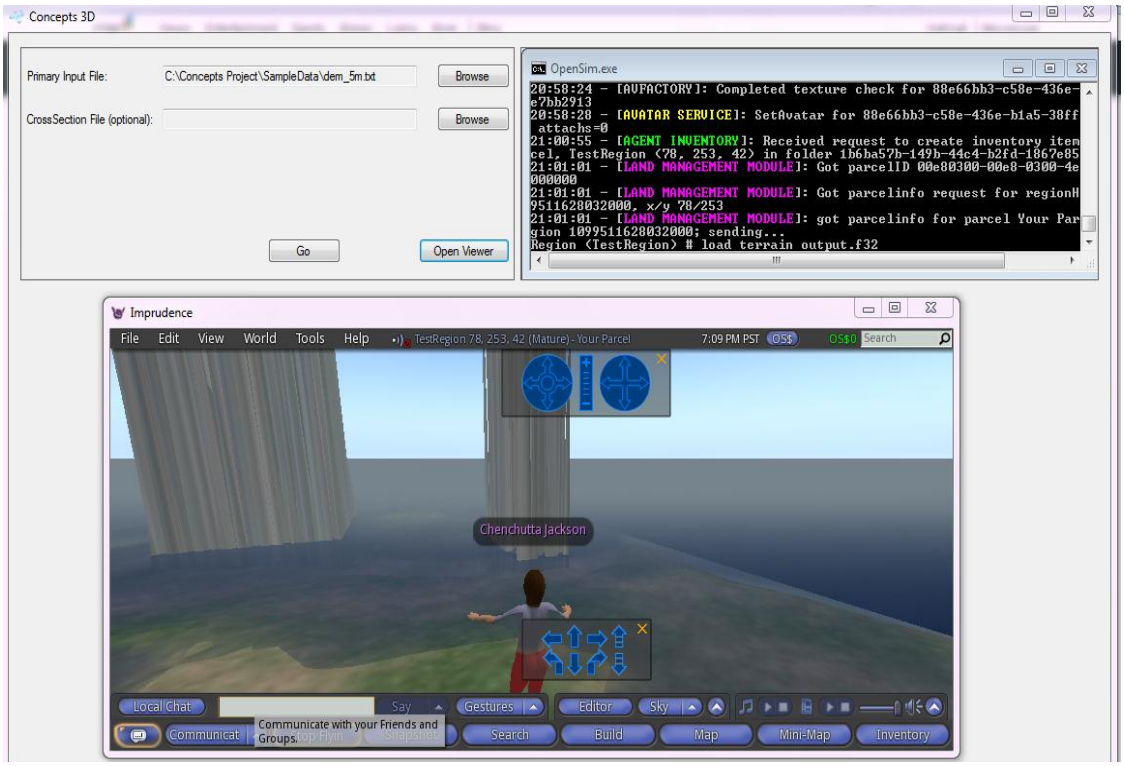


Figure 35: Results of Additional Data Points within the 5 meter DEM terrain

To correct this problem, in the data where the values are -99999 these values are set to the lowest elevation value within the data set creating a smoother edge to visualize. This resulted in an image that appears as a cliff for the edge of the data which is more realistic terrain once visualized. (See Figure 36).

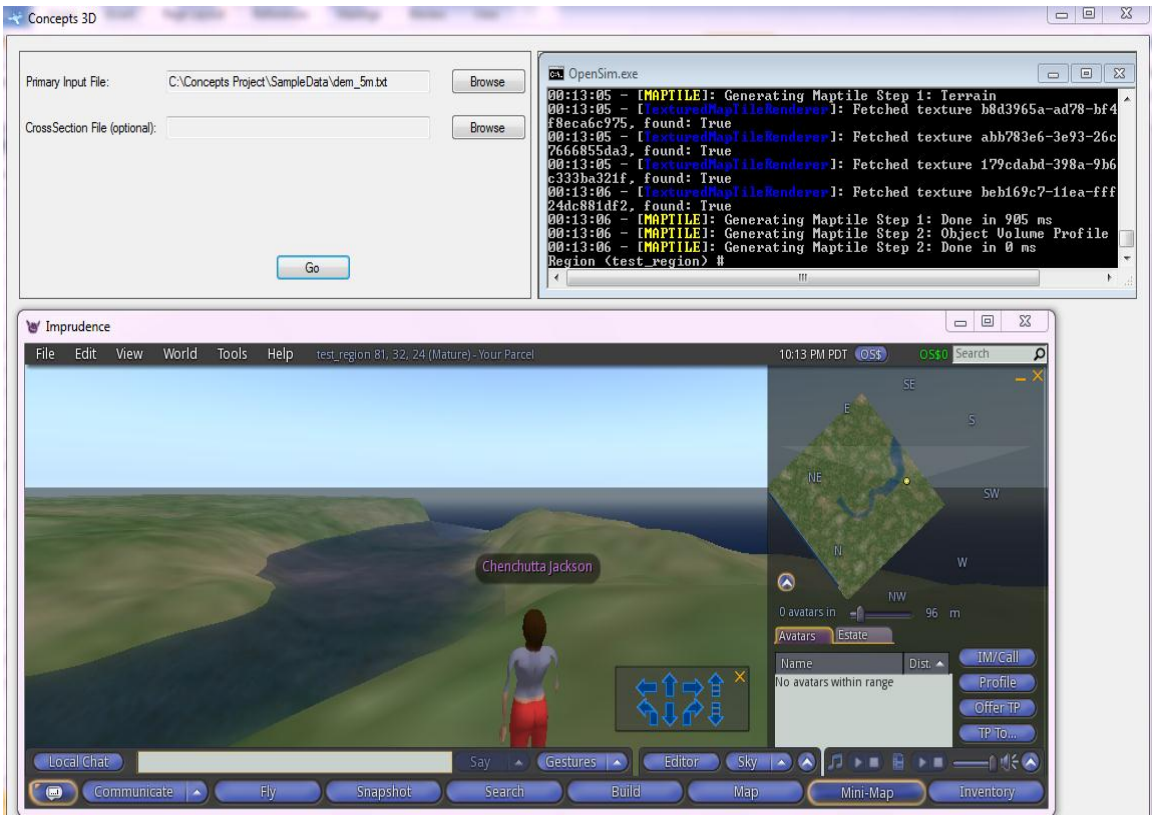


Figure 36: Smoother Edge Terrain of DEM 5 meter Data

The next test results are from combining the 1 meter DEM file with the X-Section file. The steps to visualize the data are the same as the above-mentioned steps. The user chooses the files to merge. In this case, the files are the DEM file and the X-Section file. Once the parsing process is complete, the bitmap is created. Just as before, the user is required to choose the area to visualize as a 3D virtual terrain.

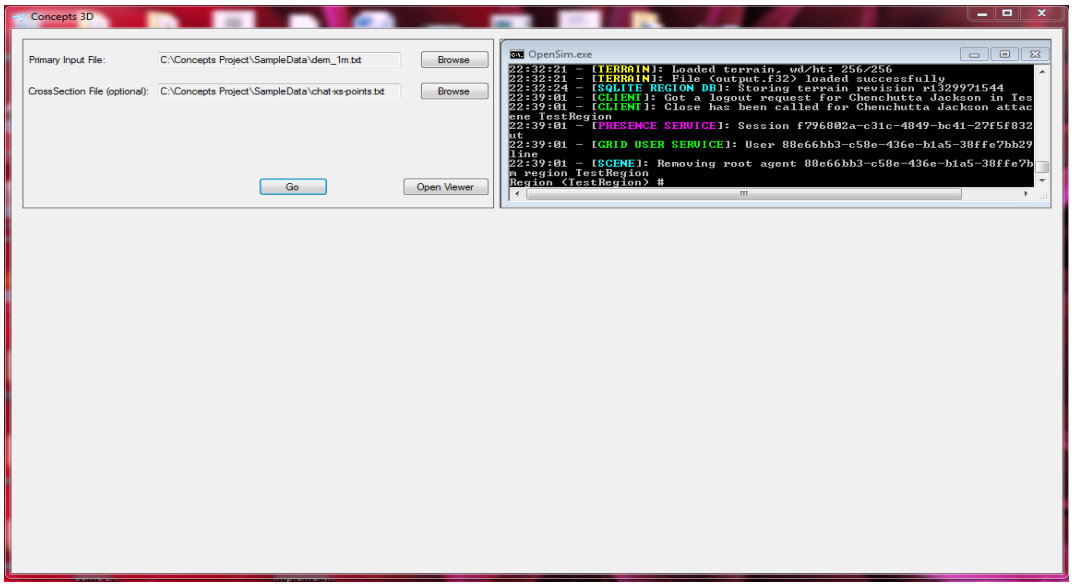


Figure 37: Dem 1 Meter File and X-Section File selected for Data Processing

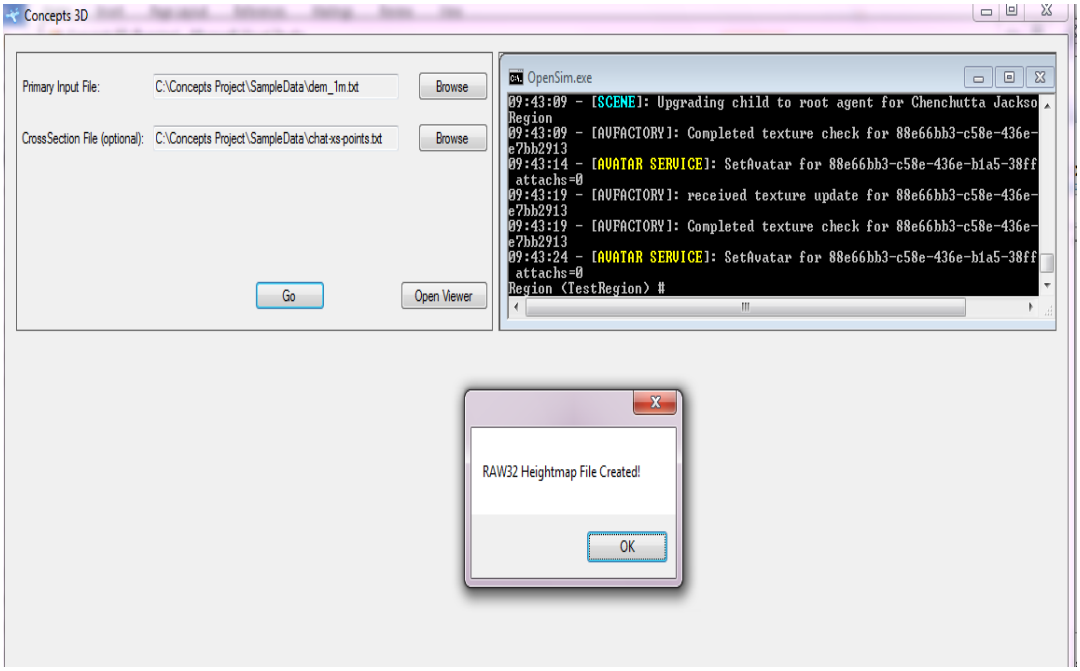


Figure 38: Raw32 File Created for the X-Section and 1 Meter Dem File

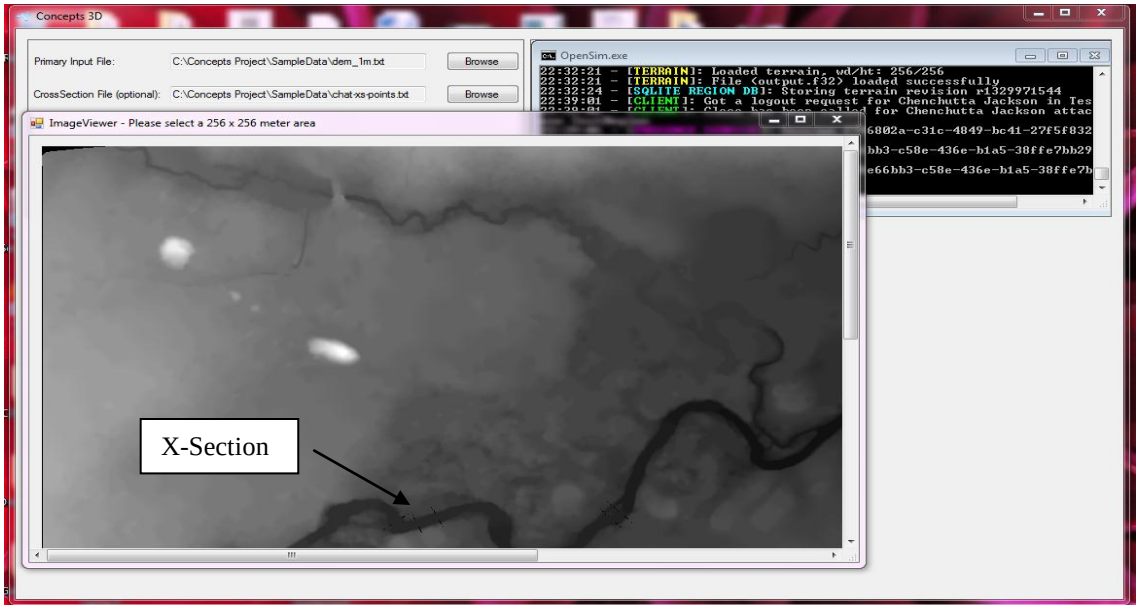


Figure 39: BitMap Image of Dem file and X-Section File, where X-Section dark lines appear within stream

The result of processing the X-Section and DEM file show in the bitmap image where the X-Section is inserted into the area. Note that lines appear darker; see figure 39 above, this is due to data mapping; the data that exists in the location is overwritten by the new data points. Results from processing X-Section and the 1 meter DEM file with the resulting virtual environment are shown in the next figures below.

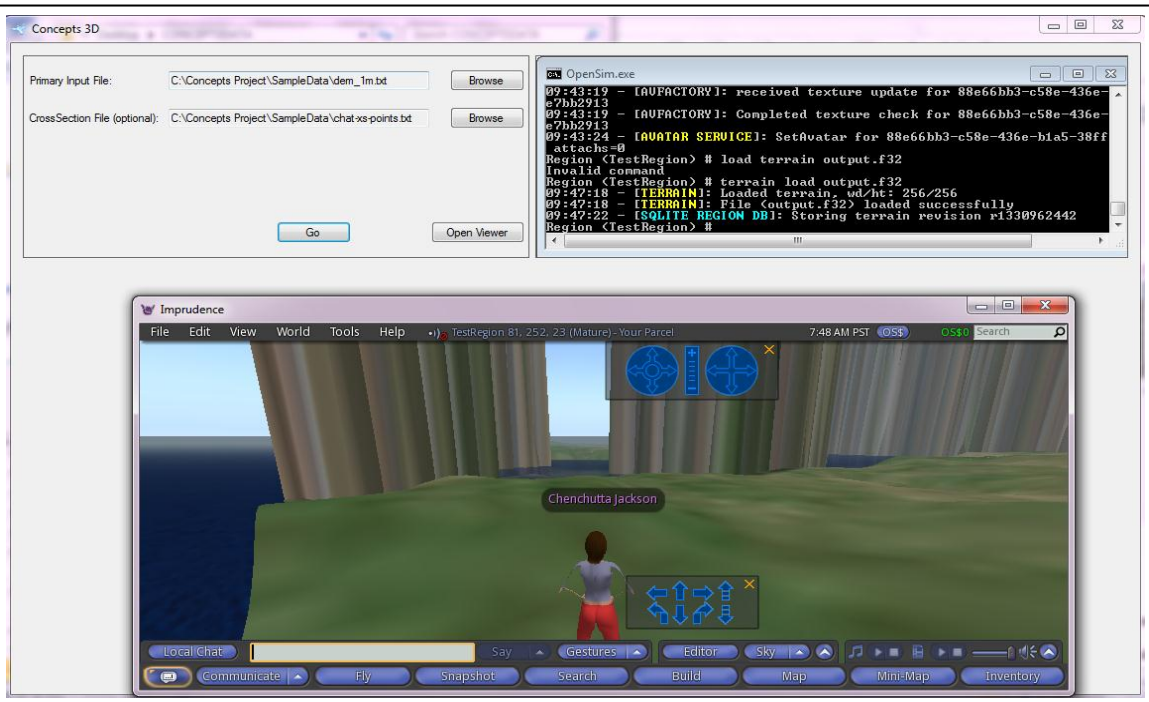


Figure 40: Result of the X-Section and Dem File Rendered

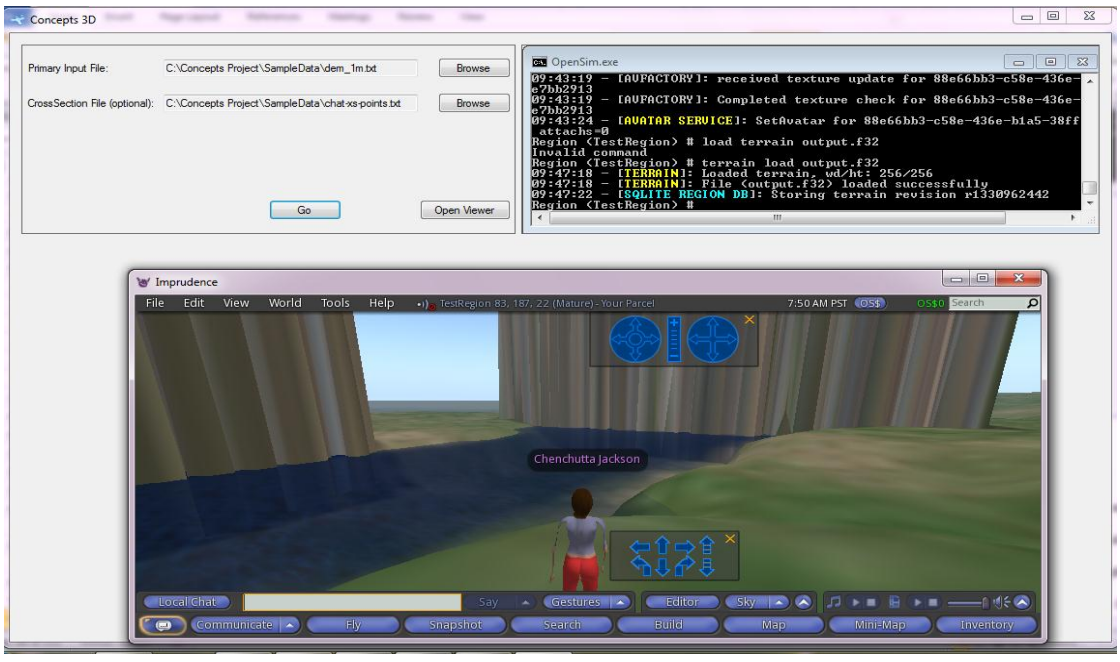


Figure 41: Display of X-Section within the Dem Terrain

To resolve the issue of huge unsightly spikes in the 1 meter DEM image with X-Section data inserted, data smoothing was done by scaling the X-Section data value with that of the corresponding floodplain data to produce a more realistic terrain. (See Figure 42)

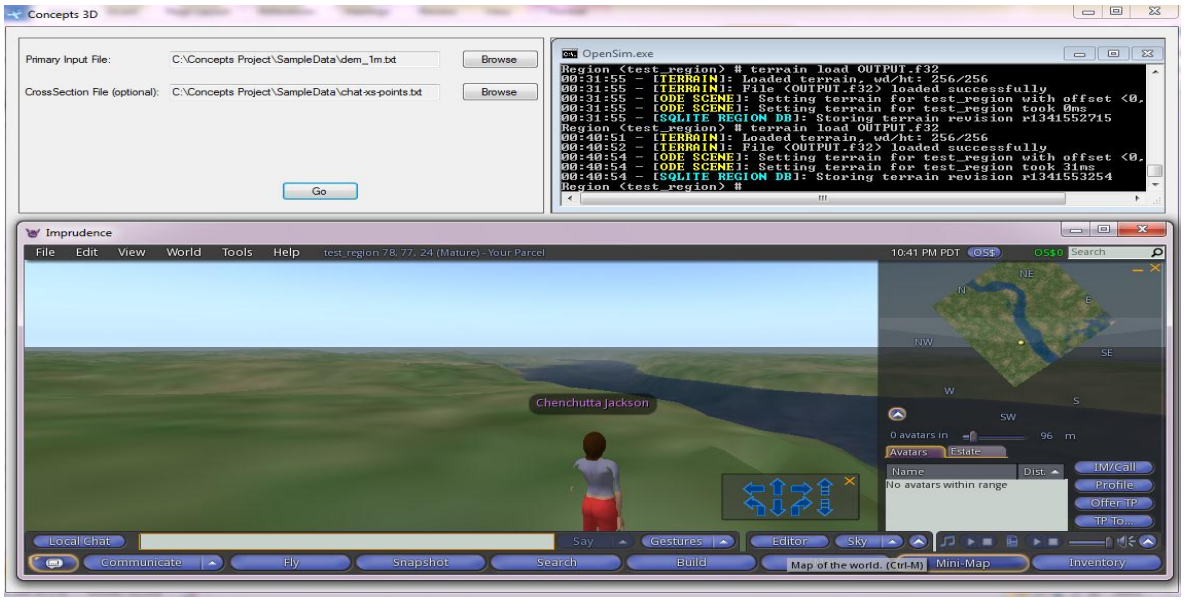


Figure 42: Improved Model with Data Smoothing

Figure 43: Bilinear Interpolation of 1 Meter Dem with X-Section after X-Section Scaling

The next figures, 44 and 45 show the results from the Barycentric interpolation method using the 1 meter data set inserted with the X-Section data.

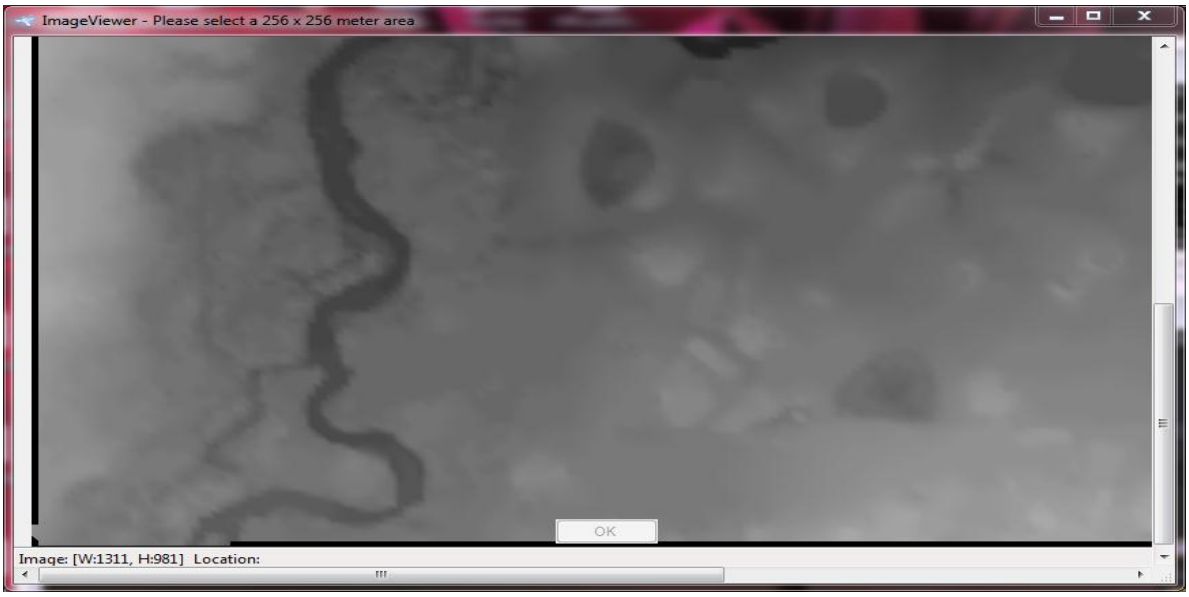


Figure 44: Barycentric Interpolation BitMap

Figure 45: Virtual Reality Terrain after Barycentric Interpolation

The exploration and testing of different gaming engines such as Neoaxis, Cafu, and OpenSim shows that the accepted file types and file sizes for each can pose problems. It has also been found for many open source engines, no true standards exist for creating graphic applications using open source applications. The types of image files with heightmap data for terrain rendering that work best for the gaming engines are various raw file formats, .png and .bmp. The smallest file size that works for most gaming engines is 256x256 pixels. For gaming engines, larger files are optimal when converted into a raw file format.

Currently, OpenSim's smallest file accepted is 256x256 pixels in any of the formats .jpg, .bmp, .png, .gif, .tif, .tiff, .r32 (RAW32), .ter (Terragen) and .raw (LL/SL RAW) (OpenSim 2011). The larger files require the data to be converted into one of the raw formats, and, even with this modification, the data must be pieced together in 256x256 tiles. The gaming engine that was chosen for this dissertation was OpenSim because it accepts a variety of data file formats, the engine performs as expected, and it is written in .NET, permitting easier software integration.

Testing the XML parsing component of the CONCEPTS3D desktop application requires the CONCEPTS XML file. Once the data is extracted from the file this information is inserted into the MySQL database CONCEPTS. Some of the one-to-many and many-to-many tables found in CONCEPTS XML were saved in a dictionary object. Saving the information in a *Collection* class allowed for faster lookup and load time and efficient data processing of large XML documents. The XML file used for testing the XML parser component was the GC_analysis CONCEPTS XML file. The results provided by executing the XML parser component is shown next. First the user is prompted to enter an XML file on the home screen of the CONCEPT3D application. The question appears because these file are large and take a long

time to load and process.

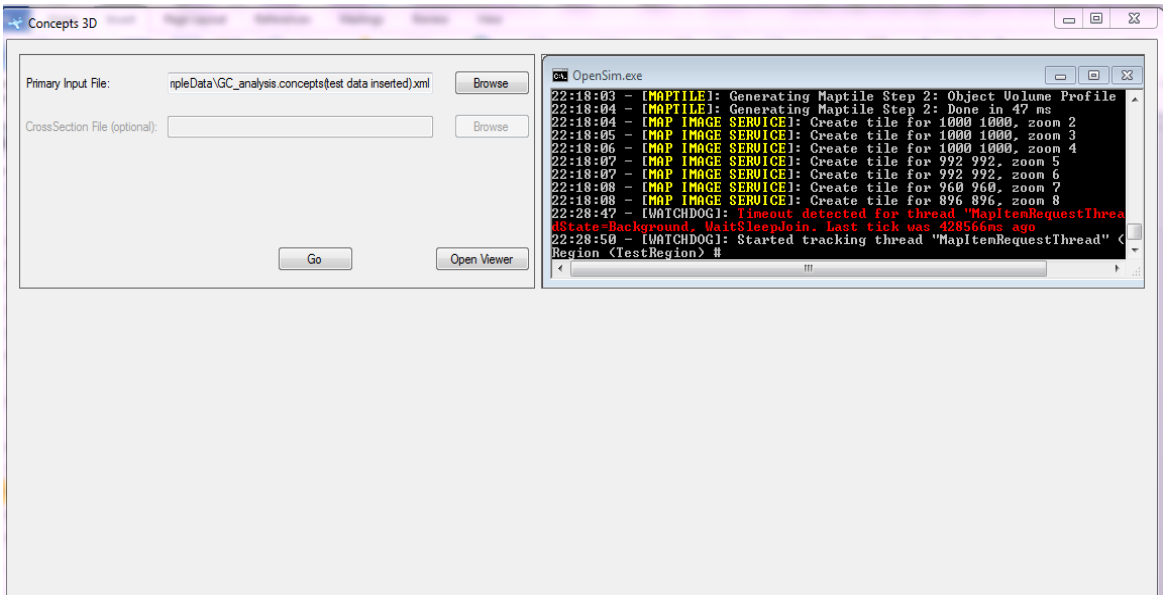


Figure 46: User Selecting XML file for Processing

The option to skip inserting into the database is given. If "Yes" is chosen inserts are performed to the database.

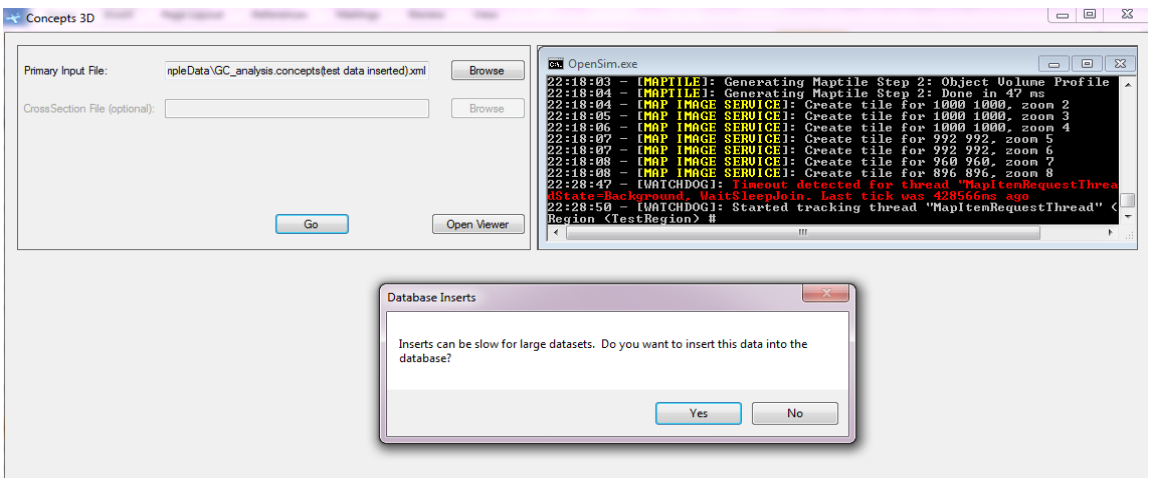


Figure 47: Message asking users to Insert Data into Database

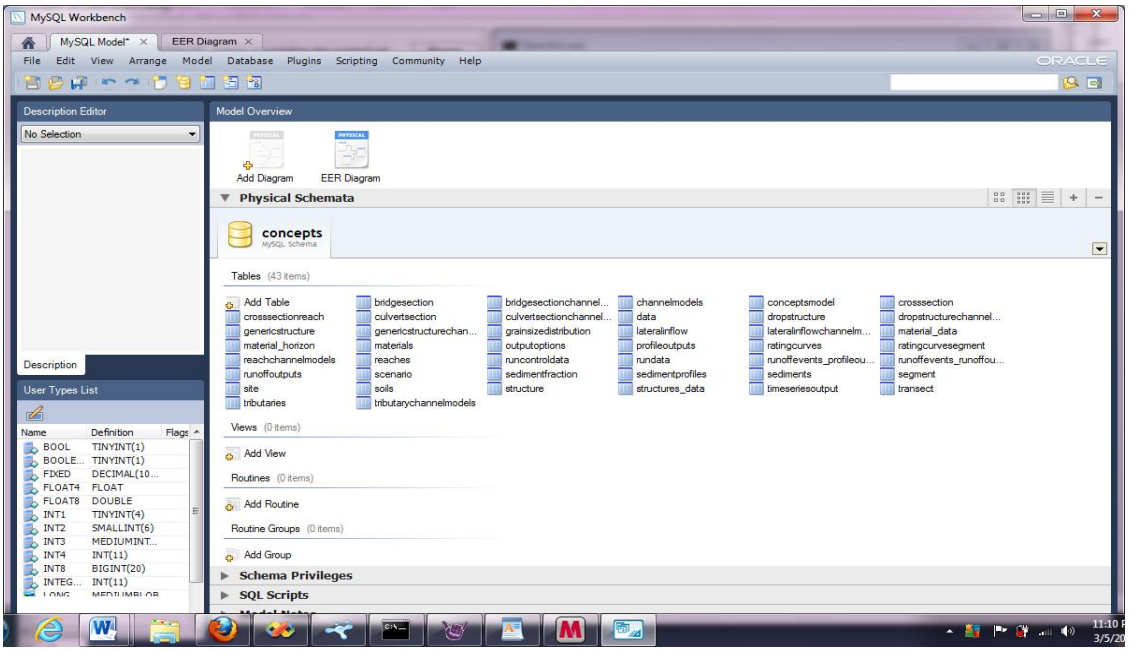


Figure 48: Concept MySQL Database

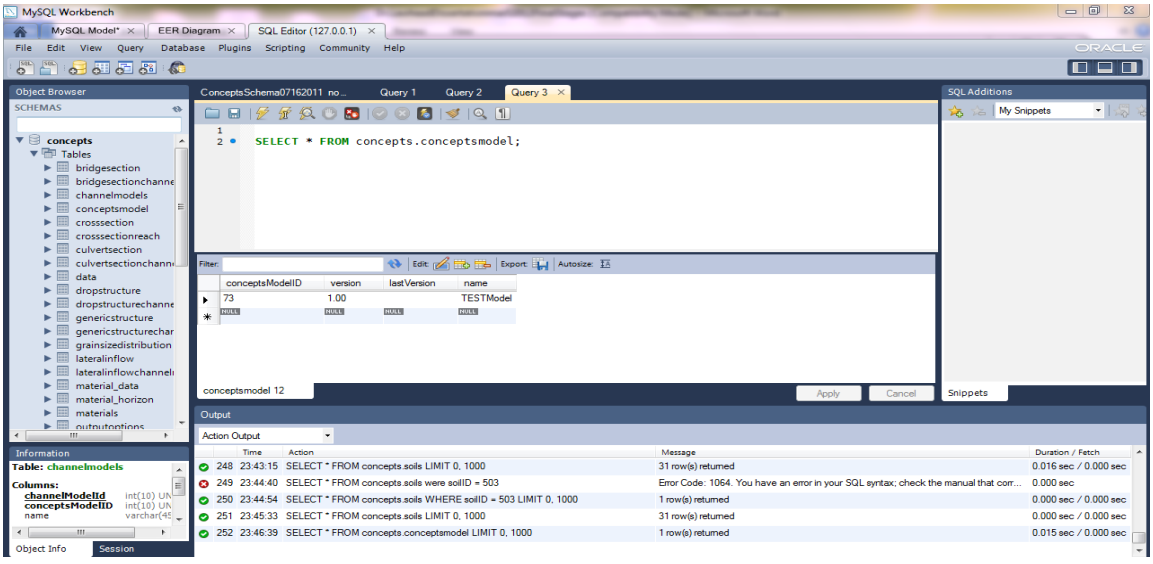


Figure 49: Select query performed to view CONCEPTS conceptsmodel table

If the user chooses “No” the application does not perform any operation and the user is placed back at the home screen of the desktop application.

Testing the GUI of the desktop application required several files, CONCEPTS 1 and 5 meter DEM files, X-Section and 1 meter DEM files, and CONCEPTS XML files. All of the files were uploaded and processed by the underlying component of the desktop application. A minor challenge exists where the gaming engine does not start in the pane of the application this is due to the thread and processing timing. Executing the component is not an issue because the user views this component as a separate window. Essentially the desktop application performs as expected. Producing the desktop application using C# .NET language and framework has made integrating components seamless. The application produced is shown below in figure 50.

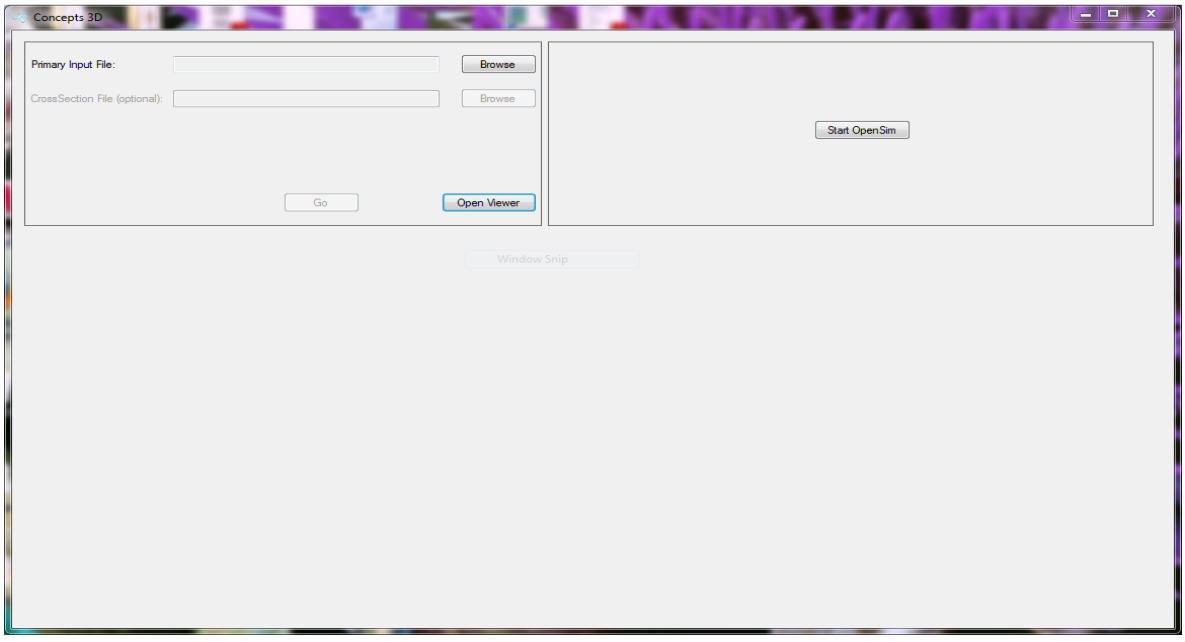


Figure 50: Prototype of CONCEPTS3D

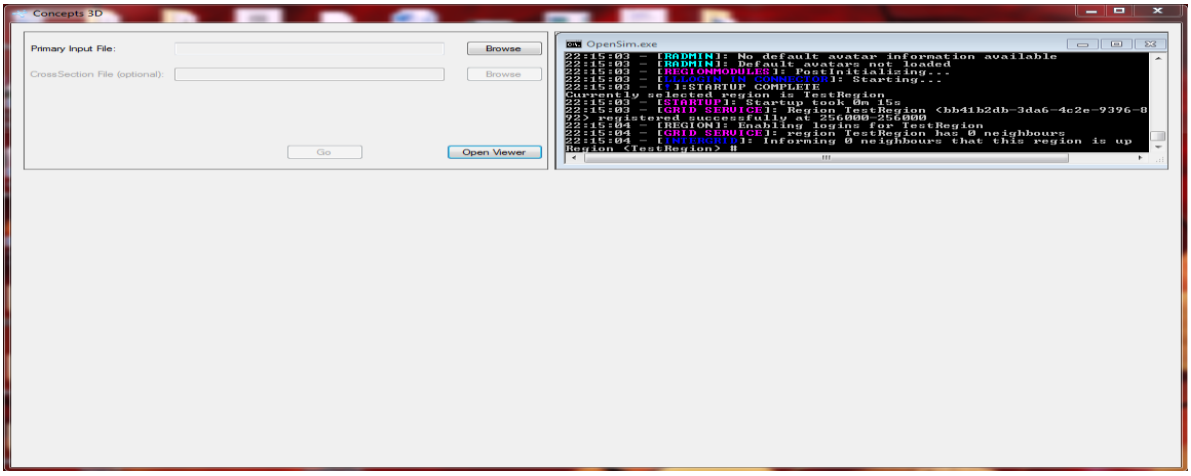


Figure 51: User starts OpenSim gaming engine

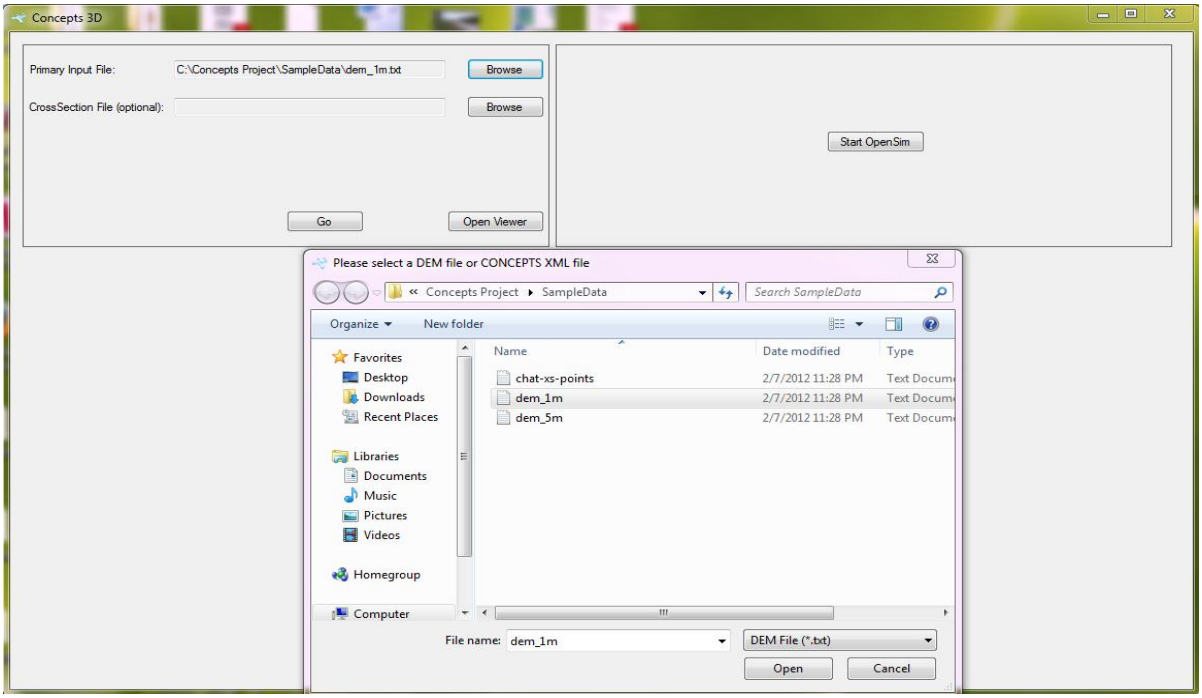


Figure 52: User select 1 meter DEM file for Processing

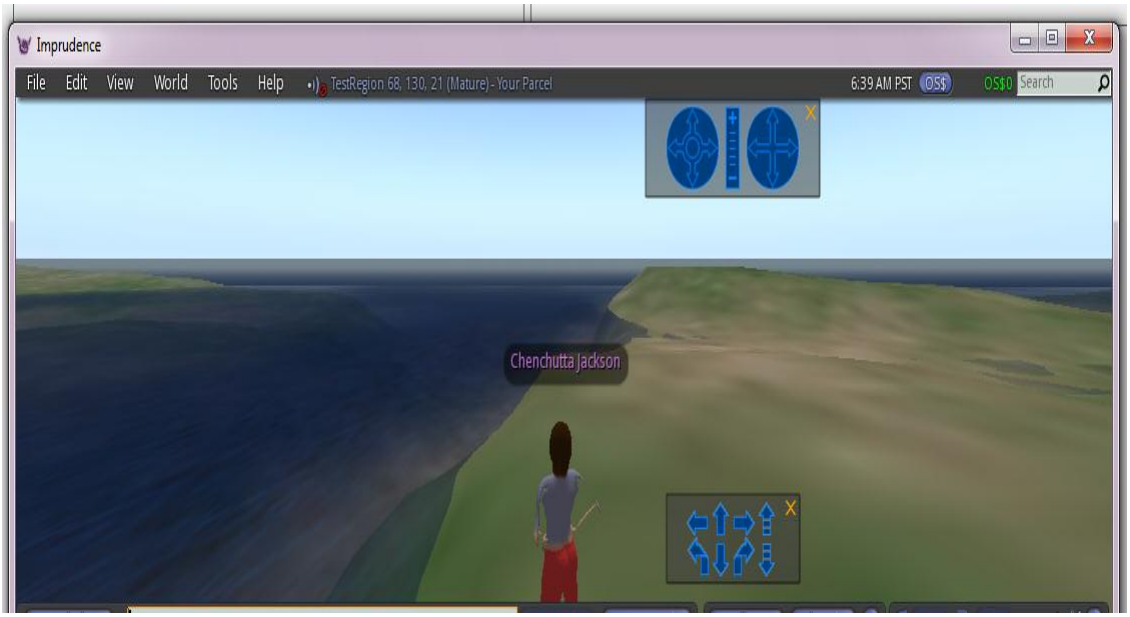


Figure 53: Result of 1 Meter DEM as a VR Terrain

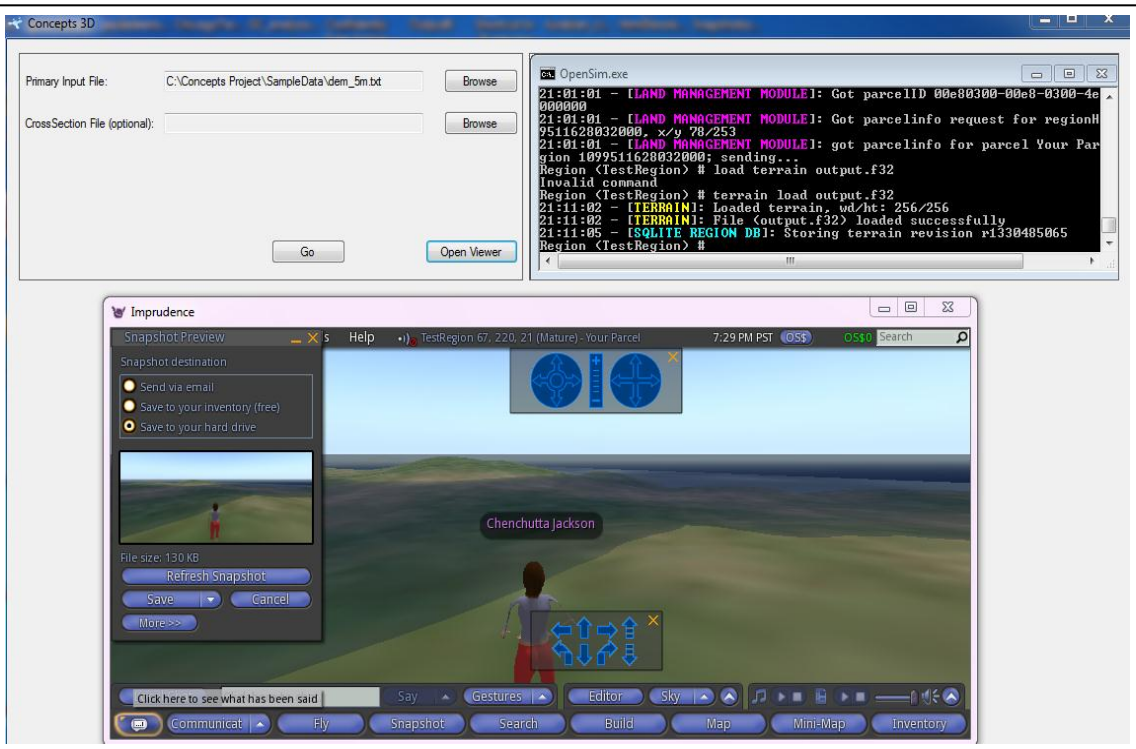


Figure 54: Results of 5 meter DEM file as a VR Terrain

CHAPTER V

CONCLUSION

The purpose of this study was to provide a bridge between two seemingly disparate worlds, the hard science of the numerical modeling of channel morphology and the rapidly developing entertainment domain of 3D game development, with a goal of using the inexpensive, though very effective, reality tools of gaming technology to visualize the complex numerical models in the channel morphology world. While this required some tweaking on both sides (standardizing the gaming engine file formats and using further mathematical modeling to expand the features of the morphology datasets), the results exceeded all expectations.

In the efforts of this dissertation there were many challenges. The obstacles of analyzing, creating, and implementing a 3D rendering desktop application was one challenge. The other major challenge was data refinement. The data that was used as input to the 3D rendering engine had to be created, modified, and enhanced in many instances. The techniques used to create, modify, and enhance the discrete data set into a continuous data set were bilinear interpolation and Delaunay triangulation with Barycentric interpolation. Results from performing these functions on the data were found to be worthwhile. Comparing the interpolation techniques, it was found that the bilinear interpolation and Barycentric interpolation both produced similarly, realistic models. While the bilinear interpolation method yielded a sharper image than Barycentric interpolation, this was to be expected since the additional data points created by

performing bilinear interpolation uses four known data points as opposed to Barycentric interpolation which only uses three. Also bilinear interpolation was an easier algorithm to implement and follow, as opposed to Barycentric interpolation which required some additional data processing before interpolation could be performed. After the dataset was enhanced with more data points and triangles, the technique of triangle cutting and data point reduction was performed by the gaming engine method of LOD. This increased the processing speed and the quality of the image rendered by the gaming engine.

The major issue of the dissertation was the data provided to visualize. This issue will forever pose a problem for both numerical models and visualization applications. The dataset used for many of the channel evolution models are of various granularities. Some datasets provided are coarser and some are fine in terms of data precision. The data that worked best for visualizing for this dissertation was data that was in meters not fractions of meters. So data precision for the X-Section dataset posed a problem. The solution was that the data is truncated. But this data did not pose a problem for visualizing the environment as a 3D virtual terrain. Because the DEM file provided a dense amount of data, because it was 1 meter, all the details of the terrain could be rendered as a 3D virtual terrain that was exact and believable.

Although it was found that the software development process for this research was progressively oscillating, the final product has proven to be beneficial for both novice and expert hydraulic engineers. Each goal articulated for this dissertation was successfully accomplished. Those objectives were:

1. Build a prototype of a desktop application available for non-sophisticated users, which visualizes CONCEPTS data in 3D-CONCEPTS3D.

2. Store and access CONCEPTS XML data through a MySQL database.

3. Develop three parsers to extract appropriate data to model a 3D virtual environment.

    a. Parser—XML file
    b. Parser—DEM file-terrain
    c. Parser—Cross-section (X-Section) file-channel

4. Transform DEM and X-Section data into a height-map file format.

5. Develop data interpolation techniques that allow for CONCEPTS data to be available for the 3D gaming engine.

6. Convert interpolated data into gaming engine native file format.

7. Find and adapt an open source gaming engine so that it can be repurposed for CONCEPTS viewing.

The rendering of the 3D virtual terrain was made possible by using the open source gaming engine OpenSim. The components within this system architecture are provided in the appendix page 122 of this dissertation.

Future Work

For the future, this dissertation is easily adaptable for an enhanced model by adding features to insert objects within the terrain. Once the terrain is created the user can modify the image by adding objects to the scene. These objects must also be created for the CONCEPTS/CONCEPTS3D model. For this project default objects could be created such as vegetation, sediment, culvert, and cobbles. These objects can be chosen from a dropdown menu and added to the scene. From here the user can re-execute the CONCEPTS model with the new conservation measures and see how these additions will affect the channel. The image of each

executed scenario is viewed and compared to see which proposed stream restoration solution will be a potential success.

For future endeavors improvement could be made to increase the usability as well as the efficiency of the computing and visualization process. To improve the software interface and infrastructure, suggestions to create components to communicate and integrate seamlessly could be performed. One component that could be integrated into the system infrastructure is the CONCEPTS simulation model. Instead of developing this component as a standalone project executing on the users machine, this component could be integrated into the 3D rendering software.

From a programming perspective, computing efficiency could be obtained by using techniques such as code placement [Tomiyama & Yasuura, 1997]. Code placement techniques are performed to reduce the missed rates of instruction caches. From a data storage and retrieval perspective the other techniques that could be used to improve performance is image caching. Another suggestion to increase performance of processing and visualizing the data, is using the GPU to process the data and off load many of the graphics handling tasks from the CPU. The final suggestion, which is one obtained to create the entire application as a web application. Allowing the user to run both CONCEPTS and CONCEPTS3D from a web browser will enhance both accessibility and simultaneous use of the application.

An advanced application is easily obtainable from this dissertation. The entire system could be developed as a web application that will allow remote access to CONCEPTS data and CONCEPT3D. Online communication would be an enhanced feature which will allow scientist to communicate with others using the system through the chat feature of the web application

version. Each integrated component will work independently but will be presented as one seamless application. This could also be made possible by using HTML5 [Chan, Holznagel, and Krantz, 2010].

BIBLIOGRAPHY

BIBLIOGRAPHY

"3D Graphics." MorpZone. http://library.morphzone.org/3D_graphics (accessed December 2011).

Angel, Edward. "Teaching a Three-Dimensional Computer Graphics Class Using OpenGL." *Computer Graphics*. (1997): 54-55.

Bash, Jeffrey, and Clare Ryan. "Stream Restoration and Enhancement Projects: Is Anyone Monitoring?." *Environmental Management*. 29.6 (2002): 877-885.

Bernhardt, Emily, Elizabeth Sudduth, Margaret Palmer, David J. Allan, Judy Meyer, Grethchen Alexander, Jennifer Follastad-Shah, Brooke Hassett, Robin Jenkinson, Rebecca Lave, Jeanne Rump, and Laura Pagano, . "Restoring Rivers One Reach at a Time: Results form a Survey of U.S. River Restoration Practitioners." *Restoration Ecology*. 15.3 (2007): 482-493.

Carsten Fuchs Software, "Welcome to Cafu." *Cafu Engine*. 2010. Web. 25 April 2010. <http://www.cafu.de/>.

Chan, Min, Fritz Holznagel, and Michael Krantz. *20 Things I learned about Browsers and the Web*. Google Chrome Team, 2010. http://www.20thingsilearned.com/media/ (accessed December 2011).

Cheng, Ge, Fei, Guo, Wang shu, Zhao Zhang, Yu Han. “3D Visualization of Pollutant Dispersion in Taihu Lake.” IEEE: 2010 2nd Conference on Environmental Science and Information Application Technology (2010)

Chybicki, Andrej, Marcin Kulawiak, Zbigniew Lubniewski, Jacek Dabrowski, Mariusz Luba, Marek Moszynski and Andrzej Stepnowski. "GIS for Remote Sensing, Anaysis and Visualizsation of Marine Pollutution and Other Marine Ecosystem Components." *IEEE*: Proceeding of the 2008 1st International Conference on Information Technology (2008): 19-21

Croix, John, and Sunil Khatri. "Introduction to GPU Programming for EDA." *IEEE/ACM International Conference on Computer-Aided Design Digest of Technical Papers* (2009): 276-280. Web. 22 February 2010.

"CUDA in Action." NVIDIA Corporation 2010: 1. Web. 22 February 2010. <www.nvidia.com>.

Daintith John. "Delaunay triangulation." A Dictionary of Computing. 2004. *Encyclopedia.com.* 21 Apr. 2012 <http://www.encyclopedia.com>.

Davison, Andrew. “Game Programming with Java and Java 3D.” O’Reilly Media (2005): 1008

Davison, Andrew. "Chapter 1. Why Java for Games Programming? The Myths (and Truths) of Java Games Programming.". N.p., March 12, 2012. Web. 21 Oct 2010. <http://fivedots.coe.psu.ac.th/~ad/jg/ch00/index.html>.

Day, Bill. "3D Graphics Programming in Java, Part 3: OpenGL." InfoWorld JavaWorld. http://www.javaworld.com/javaworld/jw-05-1999/jw-05-media.html (accessed May 2010).

"DimensioneX." http://www.dimensionex.NET/en/.August 2009. Web.

Dong, Chen, Fengyuan Liu, Haiyan Wang and Feixiang Chen. "Application Research of Mobile GIS in Forestry Informatization." *IEEE: The 5th International Conference of Computer Science & Education* (2010): 351-355.

Feibush, Eliot, Nikhil Gagvani, and Daniel Williams. "Geo-Spatial Visualization for Situation Awareness." *IEEE: Computer Graphics and Applications*. (1999): 441-443.

Fleming, Bill. *3D Modeling & Surfacing.* Academic Press. *1999*

Fishwick, Paul . "An Introduction to OpenSimulator and Virtual Environment Agent-Gased M & S Applications." *IEEE: Proceeding of the 2009 Winter Simulation Conference* (2009): 177-183.

Floriani, Leila, and Paola Magillo. "Regular and Irregular Multi-Resolution Terrain Modelings: a Comparison." *ACM*. (2002): 143-148. Print.

Friesen, Jeff. "Open Source Java Projects: Java Binding for OpenGL (JOGL)." InfoWorld JavaWorld. http://www.javaworld.com/javaworld/jw-09-2008/jw-09-opensourcejava-jogl.html (accessed May 2010).

Gangolly, Jagdish. "Software Development Lifecycle." http://www.albany.edu/acc/courses/acc681.fall00/681book/node25.html (accessed October 2009).

GeekLog, "Delta3D: Open Source Gaming and Simulation Engine." *Delta3D*. GNU Lesser General Public License, 02 May 2010. Web. 2 May 2010. <http://www.delta3d.org/>.

Glaeser, and Stachel Hellmuth. Open Geometry: OpenGL + Advanced Geometry. Springer, 1999.

Greenwood, Philip, Jesse Sago, Sam Richmond and Vivi Chau. "Using Game Engine

Technology to Create Real-time Interactive Environments to Assist in Planning and Visual Assessment for Infrastructure." *18th World IMACS/MODSIM Congress* (2009): 2229-2235.

Grevera, George and Jayaram Udupa. "An Objective Comparison of 3-D Image Interpolation Meithos." *IEEE Transactions On Medical Imaging* 17 (1998): 642-652.

Guo, Fei, Shu WAng, Zhuo Zhang, Linlin Zhao and Cheng Ge. "Simulation and Model Computation of Taihu Lake Based on GIS." National Natural Science Foundation of China and National Science and Technology R&D Program: (2008)

Han, Y., M. Kwon and H. Park. "Reduction of Triangles in Discretized Marching Cubes for 3D Rendering of Brain MR Images." *Proc. Intl. Soc. Mag. Reson. Med. 11* (2003)

Hand, Randall. "SecondLife Q2 Features: Mesh Import & Havok Physics." VizWorld. http://www.vizworld.com/2010/05/life-q2-features-mesh-import-havok-physics/ (accessed May 2011)

Harris, Mark. Mapping Computational Concepts to GPUs. GPU Gems 2. NVIDIA, 2005. 493-508. Print.

Harris, Mark. *Mapping Computational Concepts to GPUs*. NVIDIA, 2005. 493-502.

"Havok." http://www.havok.com/ (accessed April 21, 2012).

Hawkins, Kevin, and Dave Astle. "What is OpenGL?." *Premier Press* (2001): 1-4. Web. 8 March 2010. <http://www.developer.com>.

Henderson, Shane, and Andrew Mason. "Ambulance Service Planning: Simulation and Data Visualization." n. page. Web. 22 February 2010.

He, Yong, Xunfei Deng, Yongni Shao and Hui Fang. "Study on Farmland Information Management Model Integrated with GSM and a Web-Based GIS." *IEEE* (2006): 9371-9375.

Hodorowicz, L. "What is a Gaming engine?." *Distance Education Report*. (2006): Print.

Howard, Chang. "River Morphology and River Channel Changes." *Trans. Tianjin University*. 14. (2008): 254-262.

“Hydrographic Gauging Station Scoping Document.” Department of Natural Resources, Environment, The Arts and Sport. Northern Territory Government. http://www.nt.gov.au/nretas

"Java.NET The Source for Java Technology Collarboration." *Java.NET*. N.p., 02 May 2010. Web. 22 February 2010. <http://www.java.NET/>.

Jiejun, huang, Cui Wei, Cheng Ting, Zhan Yunjun and lI Ye. "Using GIS for Data Visualization and Processing in Pollution Sources Census." *IEEE* (2010): 710-713.

Johnson, Chris, Rob MacLeod, Steven Parker, and David Weinstein. "Biomedical Computing and Visualization Software Environments." *Communication of the ACM*. 47.11 (2004): 65-71. Print.

Kelly, Fred. "Restoration of Urban Streams: Practical Evaluation of Options for 319(h) Funded Projects." *USDA-NRCS*. (2001): 1-8. Print.

Khosronejad, Rennie, Neyshabouri, and Townsend, "3D Numerical Modeling of Flow and Sediment Transport in Laboratory Channel Bends*." Journal of Hydraulic Engineering* (2007) 1123-1134

Klein, Linda, Stephen Clayton, Richard Alldredge, and Peter Goodwin. "Long-Term Monitoring and Evaluation of the Lower Red River Meadow Restoration Project, Idaho, USA." *Restoration Ecologoy*. 15.2 (2007): 223-239.

Kondolf, Mathias G., and Elisabeth Micheli. "Forum: Evaluating Stream Restoration Projects." *Environmental Management*. 19.1 (1995): 1-15.

Kondolf, G. M., S. Anderson, R. Lave, L. Pagano, A. Merenlender and E.S. Bernhardt. "Two Decades of River Restoration in California: What Can We Learn?." *Restoration Ecology*. 15.3 (2007): 516-523.

Lake, P.S., N. Bond, and P. Reich. "Linking Ecological Theory with Stream Restoration." *Freshwater Biology*. 52. (2007): 597-615.

Langendoen, Eddy. *CONCEPTS-CONservational Channel Evolution and Pollutant Transport Model* . Research Report 16. USDA-ARS National Sedimentation Laboratory , 2000.

Langendoen, Eddy, R. Lowrance, and Andrew Simon. "Assessing the Impact of Riparian Processes on Stream Bank Stability." *Ecohydrology*. 2. (2009): 360-369.

Lewis, Michael, and Jeffrey Jacobson. "Gaming engines in Scientific Research." *Communication of the ACM*. 45.1 (2002): 27-31.

Lim, Choong-Gyoo and ByongTae Choi. "Hierarchical Triangular Patches for Terrain Rendering with Their Matching Blocks." *ACM:3rd International Conference on Digital Interactive*

Media in Entertainment and Arts (2008)

Lisle, T.E. "Overview: Channel Morphology and Sediment Transport in Steepland Streams." *Erosion and Sedimentation in the Pacific Rim*. 165. (1986): 287-297.

Love, Timothy, Chris Trumbauer and Charles Cole. "Implementation of a Pilot GIS for a Volunteer-Based Water Quality Monitoring Program."

Lu, Xiaolin. "Develop Web GIS Based Intelligent Transportation Application Systems with Web Service Technology." *IEEE* (2006): 159-.

Madsen, B. L., P. J. Boon, P. S. Lake, S. E. Bunn, and C. N. Dahm. "Ecological Principles and Stream Restoration." 1-8.

Mcallister, Keegan. Triangle Rasterization. October 2007. (accessed January 2012)

Mcatmney, Hugh, Bryan Duggan and Fredrick Mtenzi. "Using the Crytek Game Engine in the Dublin Institute of Technology."

McReynolds, Tom, and David Blythe. *Advanced Graphic Programming using OpenGL*. San Francisco: Morgan Kaufmann Publishers, 2005

Merwade, Venkatesh. "River Channel Morphology Model." *River Channel Morphology Model* n. pag. Web. 14 October 2008. <www.crwr.utexas.edu>.

Miller, Tom. *Beginning 3D Game Programming*. Indianapolis: SAMS, 2005. (accessed October 27, 2011).

Morrision, Michael. *Beginning Game Programming*. Indianapolis: SAMS, 2005. (accessed October 27, 2011).

Muller, Wolfgang, and Heidrun Schumann. "Visualization Methods for Time-Dependent Data-An Overview." *Proceedings of the 2003 Winter Simulation Conference*. (2003): 737-745.

NeoAxis Group. "NeoAxis." http://www.neoaxis.com/ (accessed 2010).

Northern Territory Government, . "Hydrographic Gauging Station Scoping Document." *Hydrographic Services-Department of Natural Resorces*. 1-12.

Palmer, M. A., E.S. Bernhardt, J.D. Allan, P.S. Lake, and G. Alexander. "Forum: Standards for Ecologically Successful River Restoration." *Journal of Applied Ecology*. 42. (2005): 208-217.

Pan, Zhigeng, Xiaochao Wei, and Jian Yang. "Geometric Model Reconstruction from Streams of DirectX 3D Application." *ACM*. (2005): 242-245.

"Panda3D-Free 3D Gaming engine." *Panda3D*. Carnegie Mellon University, 2010. Web. 23 April 2010. <http://www.panda3d.org/>.

Pazzaglia, Frank J. "Landscape Evolution Models." Development in Quaternary Science. 1571-0866.1, 2003

Phillips, Greg. "Success and Benefits of Stream Restoration." *AWRA Hydrology & Watershed Management*. 1.6 (2003): 1-3

Purcell, Alison H., Carla Friedrich, and Vincent H. Resh. "An Assessment of a Small Urban Stream Restoration Project in Northern California." *Restoration Ecology*. 10.4 (2002): 685-694.

Razafindrazaka, Faniry. "Delaunay Trainagulation Algorithm and Application to Terrain Generation." International Institue for Software Technology: African Institute for Mathematic Science (AIMS) (2009)

Rehman, Herath, and Katumi Musiake, "A Process Based Approach to Model Soil Erosion and Sediment Transport at Regional Scale: Model Structure, Modeling Strategies and Validation," GeoInformatics, vol 14, 1, (2003) 29-36

Rhyne, Theresa-Marie. "Visualization and the Large World of Computer Graphics: What Happens Out There?." *TR News* September-October 2007: 20-23. Print.

Roberson, Roy. "Recovering From Drought: Restoring Stream Banks a Big Step." *Southeast Farm Press* 1 October 2008: 19-20.

Roberts, G., S. Balakirsky and S. Foufou. "3D Reconstruction of Rough Terrain for USARSim Using a Height-map Method." ACM (2008): 259-264.

Sander, Pedro, Xianfeng Gu, Steven Gortler, Hugues Hoppe and John Synder. "Silhouette Clipping." *ACM: SIGGRAPH 2OOO* (2000)

Sangappa, Sudhir, K. Palaniappan, and Richard Tollerton. "Benchmarking Java Against C/C++ for Interactive Scientific Visualization." *ACM*. (2002): 236.

Schroeder, Will, Ken Martin, and Bill Lorensen. *The visualization toolkit*. Kitware Inc, 2004.

Scott, Stephen, and Yafei Jia. "Simulation of Sediment Transport and Channel

Morphology Change in Large River Systems." *US-China Workshop on Advanced Computational Modeling in Hydroscience & Engineering*. 1-11.

Selvakumar, Ariamalar. "Evaluating Receiving Water Improvements From Stream Restoration (Accotink Creek, Fairfax City, VA)." *EPA United States Environmental Protection Agency*. (2009): 1-2.

Shield, Douglas F., Charles Cooper, and Scott Knight. "Experiment in Stream Restoration." *Journal of Hydraulic Engineering*. 121.6 (1995): 494-502. Print.

Shields, Douglas F., Ronald Copeland, Peter Klingeman, Martin Doyle, and Andrew Simon. "Design for Stream Restoration." *Journal of Hydraulic Engineering*. 129.8 (2003): 575-584.

Simpson, Jake. "Gaming engine Anatomy 101." *ExtremeTech* 30 April 2002: 1-17. Web. 19 December 2009. <http://www.extremetech.com/print_article>.

Starr, Rich. "Stream Restoration." *Stream Restoration*. U.S. Fish and Wildlife Service, n.d. Web. 21 January 2010. <www.fws.gov/ChesapeakeBay/stream.html>.

Stowell, Timothy, and Brett Shelton. "Challenges, Frustrations and Triumphs of Remixing an Open Source Gaming Engine for Educational Purposes."*TechTrends*. 52.5 (2008): 32-37.

Sudduth, Elizabeth, Judy Meyer, and Emily Bernhardt. "Stream Restoration Practices in the Southeastern United States." *Restoration Ecology*. 15.3 (2007): 573-583.

"Sybase ." *Sybase History*. Sybase, 2010. Web. 12 February 2010. <http://www.sybase.com/about_sybase/history>.

Tomiyama, Hiroyuki and Hiroto Yasuura. "Code Placement Techniques for Cache Miss Rate Reduction." *ACM: ACM Transactions on Design Automation of Electronic Systems* 2 (1997): 410–429.

Trattner, Christph, Steurer, Michael, & Kappe, Frank. "Socializing Virtual Worlds with Facebook." *MindTrek* October 6th-8th, 2010: *ACM*, 2010.

"OpenGL: The Industry's Foundation for High Performance Graphics." Kronos Group", 1 May 2010. Web. <http://www.opengl.org>.

"OpenSim." http://opensimulator.org/wiki/Main_Page (accessed October 2011). ("OpenSim,")

Osterkamp, Waite, Janet Hooke, and John Ridgway. "Stream Channel Morphology and Position." *GeoIndicators* (2006): n. pag. Web. 26 Apr 2010. <www.lgt.lt/geoin>.

Palmer, Margaret, David J. Allan, Judy Meyer, and Emily Bernhardt. "River Restoration in the Twenty-First Century: Data and Experiential Knowledge to Inform Future Efforts." *Restoration Ecology*. 15.3 (2007): 427-481.

The Federal Interagency Stream Restoration Working Group. "Stream Corridor Restoration: Principles, Processes, and Practices." (1998 Revised 2001):

"Unity." Unity Technology. http://unity3d.com/unity/engine/ March 2010).

US Army Corps of Engineers, . "HEC-6." *HEC History*. N.p., n.d. Web. 12 October 2009. <http://www.hec.usace.army.mil/>.

US Army Corps of Engineers, *River hydraulics*.18. New York: American Society of Civil Engineers, 1996.

Vance, Tiffany, Sharon Mesick, and Scott Cross. "Integrating particle Tracking Models into a GIS for Analysis and Display of Environmental Phenomena." *ACM GIS*. (2009): 460-465.

Vieira, Dalmo. "Modeling Hydrodynamics, Channel Morphology, and Water Quality Using CCHE1D." *US-China Workshop on Advanced Computational Modeling in Hydroscience & Engineering*.

Vieira, Dalmo and Wei Ming Wu. "CCHE1D Version 3.0: Model Capabilities and Application." *Technical Report NCCHE-TR-2002-2005* (2002)

Wagner, Iwona, Jiri Marsalek, John McCullah, Donald Gray, Hui Xiong, Nicole Silk, Kristine Ciruna, Zhen-Gang Ji, International Research, Michael Clar, American Committee, Timothy Randhir, John Randolph, and Wei Ji. *Wetland and water resource modeling and assessment*. CRC Press, 2008.

Wang, Sam, and Yafei Jia. "A Discussion on the Computational Modeling of Channel Morphology."

Weber, Joseph. "ProteinShader: Illustrative Rendering of Macromolecules." *BMC Structural Biology*. 9.19 (2009): 1-19.

Wegner, Katja. "SIMWIZ3D-Visualising biochemical simulation results." *IEEE* (2005) (Wegner 2005)

Wells, William. "Generating Enhanced Natural Environments and Terrain for Interactive Combat Simulations (GENETICS)." *ACM* (2005): 184-191.

Willis, B.G. Krishnapan, “Numerical Modeling of Cohesive Sediment Transport in Rivers”, Can. J. Civ. Eng. 31 (2004) 749-758

Wilson, Leslie, and Robert Clark. *Comparative Programming Languages*. 3rd ed. Great Britain: Addison-Wesley Publishers Limited, 2001.

"What is CUDA?." *CUDA GPUs*. 2010. NVIDIA, Web. 24 Feb 2010. <http://www.nvidia.com/object/what-is_cuda_new.html>.

Wolff, David. "Using OpenGL in Java with JOGL." *Consortium for Computing Sciences in Colleges*. (2005): 223-224.

Woolsey, Sharon, Florence Capelli, Tom Gonser, Eduard Hoehn, and Markus Hostmann. "A Strategy to Assess River Restoration Success." *Freshwater Biology*. 52. (2007): 752-769.

Yang, Chi, Franciso Simoes, Jianchun Juang, and Blair Greimann. "Generalized Sediment Transport Models for Alluvial Rivers and Reservoirs." *US-China Workshop on Advanced Computational Modeling in Hydroscience & Engineering*.

Yee, Jerry. "A Survey of Graphic Programming Languages." 1-10.

Yoon, and Seung-Kyu Kang, “A Numerical Model of Sediment-laden Turbulent Flow in an Open Channel.” NRC 32 (2005): 233-240

Zutshi, Aditya and Geetika Sharma. "A Study of Virtual Environments for Enterprise Collaboration." *ACM: VRCAI* (2009): 331-333.

APPENDIX

XMLPARSER

Written by Chenchutta Denaye Cross Jackson
October 10, 2010
Version 1
Copy write July 20, 2012
The XML parser was created to extract information from an XML file to be inserted into the MYSQL database.

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Xml;
using MySql.Data.MySqlClient;
using System.Data.SqlClient;
using System.Drawing;
using System.Windows.Forms;

namespace Concepts3D
{
    class ConceptsXMLParser
    {


            /**
             *
             * @param args
             */
            private MySqlConnection conn;
            private Dictionary <String, Int32> xsidMap; //key is xsid and
value is primary key from database

            private Boolean performInserts = true;

            public ConceptsXMLParser(String inputFile)
            {
                DialogResult result = MessageBox.Show("Inserts can be slow
for large datasets.  Do you want to insert this data into the database?",
"Database Inserts", MessageBoxButtons.YesNo);

                if (result == DialogResult.Yes)
                {
                    performInserts = true;
                }
                else
                {
                    performInserts = false;
                }
```

```
                parse(inputFile);
                //datacols = (int)(highestEasting - lowestEasting) + 1;
                //datarows = (int)(highestNorthing - lowestNorthing) + 1;
                //elevationData = new Single[datarows, datacols];

                //int listIndex = 0;
                //for (int y = 0; y < datarows; y++)
                //{
                //    for(int x = 0; x < datacols; x++)
                //    {
                //        elevationData[y, x] =
pointList[listIndex++].getZ();
                //    }
                //}

            }

            private void connectToDB()
            {
                //conn = new
MySqlConnection("jdbc:mysql://localhost:3306/concepts?user=root&password=");
                conn = new
MySqlConnection("SERVER=localhost;DATABASE=concepts;UID=root;PASSWORD=;");

            }

            public String parse(String inputFile)
            {
                String result = "";
                FileStream input = null;
                try
                {

                    this.connectToDB();

                    conn.Open();
                    xsidMap = new Dictionary<String, Int32>();

                    XMLLoader parser = new XMLLoader();

                    Document xmlDoc = parser.build(inputFile);

                    // get root element and parse it
                    Element conceptsModel = xmlDoc.getRootElement();

                    Console.WriteLine("Parsing concepts xml file....");

                    this.parseConceptsModel(conceptsModel);

                    Console.WriteLine("Done!");
                    result = "Content parsed and inserted into the
database.";
```

```
            }
            catch (Exception e)
            {
                // TODO Auto-generated catch block
                Console.WriteLine("Exception: " + e.StackTrace);
                MessageBox.Show("Exception: " + e.StackTrace);
                result = "An error occurred!";
            }

            if (conn != null)
                conn.Close();

            if (input != null)
                input.Close();

            return result;
        }


        private void parseConceptsModel(Element conceptsModel)
        {

            // COLUMNS SECTION

            String versionText = this.parseString(conceptsModel
                    .getAttributeValue("version"));
            String lastVersionText = this.parseString(conceptsModel
                    .getAttributeValue("lastversion"));

            String name = "TESTModel";

            // DB INSERT SECTION
            String query = "INSERT INTO conceptsmodel (name, version,
lastVersion) "
                    + "VALUES ('" + name + "'," + versionText + ", " +
lastVersionText + ")";
            int conceptsModelID = this.executeDBUpdate(query);

            // FOREIGN KEY SECTION
            // get data element and parse it
            Element data = conceptsModel.getChild("data");
            this.parseData(conceptsModelID, data);

            // actually get children of channelModels and parse each one
of them
            Element channelModels =
conceptsModel.getChild("channelModels");
            List<Element> channelModelList = channelModels
                    .getChildren("channelModel");

            // for (Element channelModel in channelModelList) {
            // this.parseChannelModel(conceptsModelID, channelModel);
            // }
            // Many to many relationships must be handled differently
```

```
                for (int index = 0; index < channelModelList.Count; index++)
                {
                    this.parseChannelModel(conceptsModelID,
channelModelList[index], index);
                }

                // get runData element and parse it
                Element runData = conceptsModel.getChild("runData");
                this.parseRunData(conceptsModelID, runData);
            }

            private void parseData(int conceptsModelID, Element data)
            {

                String query = "INSERT INTO data (conceptsModelID) " +
"VALUES ("
                        + conceptsModelID + ")";
                int dataID = this.executeDBUpdate(query);

                Element materials = data.getChild("materials");
                this.parseMaterials_Data(dataID, materials);

                Element crossSections = data.getChild("crossSections");
                List<Element> crossSectionList = crossSections
                        .getChildren("crossSection");
                for (int i = 0; i < crossSectionList.Count; i++)
                {
                    this.parseCrossSections(dataID, crossSectionList[i], i);
                }

                Element reaches = data.getChild("reaches");
                List<Element> reachList = reaches.getChildren("reach");
                for (int i = 0; i < reachList.Count; i++)
                {
                    this.parseReaches(dataID, reachList[i], i);
                }

                Element structures = data.getChild("structures");
                this.parseStructuresData(dataID, structures);

                Element tributaries = data.getChild("tributaries");
                List<Element> tributaryList =
reaches.getChildren("tributary");
                for (int i = 0; i < tributaryList.Count; i++)
                {
                    this.parseTributaries(dataID, tributaryList[i], i);
                }

                Element lateralInflows = data.getChild("lateralInflows");
                List<Element> lateralInflowList = lateralInflows
                        .getChildren("lateralInflow");
                for (int i = 0; i < lateralInflowList.Count; i++)
                {
```

```
                    this.parseLateralInflows(dataID, lateralInflowList[i],
i);
                }

            }

            private void parseChannelModel(int conceptsModelID, Element
channelModel,
                    int index)
            {

                String nameString = this.parseString(channelModel
                        .getAttributeValue("name"));

                String query = "INSERT INTO channelmodels (conceptsModelID,
name) "
                        + "VALUES (" + conceptsModelID + ", " + nameString +
")";
                int channelModelID = this.executeDBUpdate(query);

                mapXSIDString(channelModel, index, channelModelID);

                // INSERT MANY TO MANY RELATIONSHIPS
                // Reach-ChannelModel
                foreach (Element reach in
(List<Element>)channelModel.getChildren("reach"))
                {
                    String reachXSID = reach.getText();
                    int reachID = xsidMap[reachXSID];
                    query = "INSERT INTO reachchannelmodels (reachID,
channelModelID) VALUES ("
                            + reachID + ", " + channelModelID + ")";
                    this.executeDBUpdate(query);
                }

                // CulvertSection-ChannelModel
                foreach (Element culvertSection in
(List<Element>)channelModel
                        .getChildren("culvertSection"))
                {
                    String culvertSectionXSID = culvertSection.getText();
                    int culvertSectionID = xsidMap[culvertSectionXSID];
                    query = "INSERT INTO culvertsectionchannelmodels
(culvertSectionID, channelModelID) VALUES ("
                            + culvertSectionID + ", " + channelModelID + ")";
                    this.executeDBUpdate(query);
                }

                // BridgeSection-ChannelModel
                foreach (Element bridgeSection in (List<Element>)channelModel
                        .getChildren("bridgeSection"))
                {
                    String bridgeSectionXSID = bridgeSection.getText();
                    int bridgeSectionID = xsidMap[bridgeSectionXSID];
```

```
                    query = "INSERT INTO bridgesectionchannelmodels
(bridgeSectionID, channelModelID) VALUES ("
                            + bridgeSectionID + ", " + channelModelID + ")";
                    this.executeDBUpdate(query);
                }

                // DropStructure-ChannelModel
                foreach (Element dropStructure in (List<Element>)channelModel
                        .getChildren("dropStructure"))
                {
                    String dropStructureXSID = dropStructure.getText();
                    int dropStructureID = xsidMap[dropStructureXSID];
                    query = "INSERT INTO dropstructurechannelmodels
(dropStructureID, channelModelID) VALUES ("
                            + dropStructureID + ", " + channelModelID + ")";
                    this.executeDBUpdate(query);
                }

                // GenericStructure-ChannelModel
                foreach (Element genericStructure in
(List<Element>)channelModel
                        .getChildren("genericStructure"))
                {
                    String genericStructureXSID = genericStructure.getText();
                    int genericStructureID = xsidMap[genericStructureXSID];
                    query = "INSERT INTO genericstructurechannelmodels
(genericStructureID, channelModelID) VALUES ("
                            + genericStructureID + ", " + channelModelID +
")";
                    this.executeDBUpdate(query);
                }

                // Tributary-ChannelModel
                foreach (Element tributary in (List<Element>)channelModel
                        .getChildren("tributary"))
                {
                    String tributaryXSID = tributary.getText();
                    int tributaryID = xsidMap[tributaryXSID];
                    query = "INSERT INTO tributarychannelmodels (tributaryID,
channelModelID) VALUES ("
                            + tributaryID + ", " + channelModelID + ")";
                    this.executeDBUpdate(query);
                }

                // LateralInflow-ChannelModel
                foreach (Element lateralInflow in (List<Element>)channelModel
                        .getChildren("lateralInflow"))
                {
                    String lateralInflowXSID = lateralInflow.getText();
                    int lateralInflowID = xsidMap[lateralInflowXSID];
                    query = "INSERT INTO lateralinflowchannelmodels
(lateralInflowID, channelModelID) VALUES ("
                            + lateralInflowID + ", " + channelModelID + ")";
                    this.executeDBUpdate(query);
```

```
            }
        }

        private void parseRunData(int conceptsModelID, Element runData)
        {

            String query = "INSERT INTO rundata (conceptsModelID) " +
"VALUES ("
                    + conceptsModelID + ")";
            int runDataID = this.executeDBUpdate(query);

            Element runControlDataSets =
runData.getChild("runControlDataSets");
            List<Element> runControlDataList =
runControlDataSets.getChildren("runControlData");
            for (int i = 0; i < runControlDataList.Count; i++)
            {
                this.parseRunControlData(runDataID,
runControlDataList[i]);
            }

            Element outputOptionsSets =
runData.getChild("outputOptionsSets");
            List<Element> outputOptionsList =
outputOptionsSets.getChildren("outputOptions");
            for (int i = 0; i < outputOptionsList.Count; i++)
            {
                this.parseOutputOptions(runDataID, outputOptionsList[i]);
            }

            Element scenarios = runData.getChild("scenarios");
            List<Element> scenariosList =
scenarios.getChildren("scenario");
            for (int i = 0; i < scenariosList.Count; i++)
            {
                this.parseScenarios(runDataID, scenariosList[i]);
            }

        }


        private void parseMaterials_Data(int dataID, Element
material_data)
        {

            String query = "INSERT INTO material_data (dataID) " +
"VALUES ("
                    + dataID + ")";
            int material_DataID = this.executeDBUpdate(query);

            /*
             * TODO: sedimentProfile* soilProfile*
             */
            Element sediments = material_data.getChild("sediments");
```

```
                this.parseSediments(material_DataID, sediments);

                Element soils = material_data.getChild("soils");
                List<Element> soilsList = soils.getChildren("soil");
                for (int index = 0; index < soilsList.Count; index++)
                {
                    this.parseSoil(material_DataID, soilsList[index], index);
                }

                Element sedimentProfiles =
material_data.getChild("sedimentProfiles");
                List<Element> sedimentProfileList =
sedimentProfiles.getChildren("sedimentProfile");
                for (int index = 0; index < sedimentProfileList.Count;
index++)
                {
                    this.parseSedimentProfile(material_DataID,
sedimentProfileList[index], index);
                }

                //Soil profile is the same type as sediment profile so we
treat them the exact same
                //and we use the same parseSedimentProfile method
                Element soilProfiles =
material_data.getChild("soilProfiles");
                List<Element> soilProfileList =
soilProfiles.getChildren("soilProfile");
                for (int index = 0; index < soilProfileList.Count; index++)
                {
                    this.parseSedimentProfile(material_DataID,
soilProfileList[index], index);
                }


            }

            private void parseSediments(int material_DataID, Element
sediments)
            {
                // TODO Auto-generated method stub


                List<Element> materialsList =
sediments.getChildren("material");

                for (int index = 0; index < materialsList.Count; index++)
                {
                    Element materials = materialsList[0];
                    int materialID = this.parseMaterials(materials, index);
                    String query = "INSERT INTO sediments (material_DataID,
materialID) " + "VALUES (" + material_DataID + ", " + materialID + ")";
                    this.executeDBUpdate(query);
                }
```

```
            }

            private void parseSoil(int material_DataID, Element soil, int
index)
            {

                Single? bulkDensity =
this.parseSingle(soil.getChildTextTrim("bulkDensity"));
                Single? permeability =
this.parseSingle(soil.getChildTextTrim("permeability"));

                //Soil element is purposefully passed into parseMaterials
method because it is an extension of parseMaterials
                int materialID = this.parseMaterials(soil, index);

                String query = "INSERT INTO soils (material_DataID,
materialID, bulkDensity, permeability) " + "VALUES (" + material_DataID + ",
" + materialID + ", " + bulkDensity + ", " + permeability + ")";
                int soilID = this.executeDBUpdate(query);

                //mapXSIDString(soil, index, soilID);
            }

            private void parseSedimentProfile(int material_DataID, Element
sedimentProfile, int index)
            {
                // TODO Auto-generated method stub
                String id =
this.parseString(sedimentProfile.getAttributeValue("id"));
                String query = "INSERT INTO sedimentProfiles
(material_DataID, id) " + "VALUES (" + material_DataID + ", " + id + ")";
                int sedimentProfileID = this.executeDBUpdate(query);

                Element site = sedimentProfile.getChild("site");
                this.parseSite(sedimentProfileID, null, site);

                Element materialHorizon =
sedimentProfile.getChild("materialHorizon");
                this.parseMaterialHorizon(sedimentProfileID,
materialHorizon);

                mapXSIDString(sedimentProfile, index, sedimentProfileID);

            }

            private void parseMaterialHorizon(int sedimentProfileID, Element
materialHorizon)
            {
                Single? topDepth =
this.parseSingle(materialHorizon.getChildTextTrim("topDepth"));
                Single? bottomDepth =
this.parseSingle(materialHorizon.getChildTextTrim("bottomDepth"));
```

```
                String materialXSID =
materialHorizon.getChildTextTrim("material");
                int materialID = xsidMap[materialXSID];

                String query = "INSERT INTO material_horizon (topDepth,
bottomDepth, materialID, sedimentProfileID) VALUES (" + topDepth + ", " +
bottomDepth + ", " + materialID + ", " + sedimentProfileID + ")";
                executeDBUpdate(query);
            }


            private void parseSite(int? sedimentProfileID, int? transectID,
Element site)
            {
                String id = this.parseString(site.getAttributeValue("id"));
                Single? easting =
this.parseSingle(site.getChildTextTrim("easting"));
                Single? northing =
this.parseSingle(site.getChildTextTrim("northing"));
                Single? elevation =
this.parseSingle(site.getChildTextTrim("elevation"));
                Single? station =
this.parseSingle(site.getChildTextTrim("station"));

                String query = "INSERT INTO site (id, easting, northing,
elevation, station, sedimentProfileID, transectID) VALUES " +
                                "(" + id + ", " + easting + ", " + northing +
", " + elevation + ", " + station + ", " + sedimentProfileID + ", " +
transectID + ")";

                executeDBUpdate(query);

            }

            private int parseMaterials(Element materials, int index)
            {
                // TODO Auto-generated method stub

                String name =
this.parseString(materials.getChildTextTrim("name"));
                Single? particleDensity =
this.parseSingle(materials.getChildTextTrim("particleDensity"));
                Single? porosity =
this.parseSingle(materials.getChildTextTrim("porosity"));
                Single? erodibility =
this.parseSingle(materials.getChildTextTrim("erodibility"));
                Single? criticalShearStress =
this.parseSingle(materials.getChildTextTrim("criticalShearStress"));
                Single? cohesion =
this.parseSingle(materials.getChildTextTrim("cohesion"));
                Single? frictionAngle =
this.parseSingle(materials.getChildTextTrim("frictionAngle"));
                Single? phiB =
this.parseSingle(materials.getChildTextTrim("phiB"));
```

```
                Element grainSizeDistribution =
materials.getChild("grainSizeDistribution");
                int grainSizeDistributionID =
this.parseGrainSizeDistribution(grainSizeDistribution);


                String query = "INSERT INTO materials
(grainSizeDistributionID, name, particleDensity, porosity, erodibility,
criticalShearStress, cohesion, frictionAngle, phiB) VALUES (" +
grainSizeDistributionID + ", " + name + ", " + particleDensity + ", " +
porosity + ", " + erodibility + ", " + criticalShearStress + ", " + cohesion
+ ", " + frictionAngle + ", " + phiB + ")";
                int materialID = this.executeDBUpdate(query);

                mapXSIDString(materials, index, materialID);

                return materialID;

            }

            private int parseGrainSizeDistribution(Element
grainSizeDistribution)
            {
                Single? clayTotal =
this.parseSingle(grainSizeDistribution.getChildTextTrim("clayTotal"));
                Single? siltVeryFine =
this.parseSingle(grainSizeDistribution.getChildTextTrim("siltVeryFine"));
                Single? siltFine =
this.parseSingle(grainSizeDistribution.getChildTextTrim("siltFine"));
                Single? siltMedium =
this.parseSingle(grainSizeDistribution.getChildTextTrim("siltMedium"));
                Single? siltCoarse =
this.parseSingle(grainSizeDistribution.getChildTextTrim("siltCoarse"));
                Single? siltVeryCoarse =
this.parseSingle(grainSizeDistribution.getChildTextTrim("siltVeryCoarse"));
                Single? sandVeryFine =
this.parseSingle(grainSizeDistribution.getChildTextTrim("sandVeryFine"));
                Single? sandFine =
this.parseSingle(grainSizeDistribution.getChildTextTrim("sandFine"));
                Single? sandMedium =
this.parseSingle(grainSizeDistribution.getChildTextTrim("sandMedium"));
                Single? sandCoarse =
this.parseSingle(grainSizeDistribution.getChildTextTrim("sandCoarse"));
                Single? sandVeryCoarse =
this.parseSingle(grainSizeDistribution.getChildTextTrim("sandVeryCoarse"));
                Single? gravelVeryFine =
this.parseSingle(grainSizeDistribution.getChildTextTrim("gravelVeryFine"));
                Single? gravelFine =
this.parseSingle(grainSizeDistribution.getChildTextTrim("gravelFine"));
                Single? gravelMedium =
this.parseSingle(grainSizeDistribution.getChildTextTrim("gravelMedium"));
                Single? gravelCoarse =
this.parseSingle(grainSizeDistribution.getChildTextTrim("gravelCoarse"));
```

```
                Single? gravelVeryCoarse =
this.parseSingle(grainSizeDistribution.getChildTextTrim("gravelVeryCoarse"));
                Single? cobblesSmall =
this.parseSingle(grainSizeDistribution.getChildTextTrim("cobblesSmall"));

                String query = "INSERT INTO grainSizeDistribution (clayTotal,
siltVeryFine, siltFine, siltMedium, siltCoarse, siltVeryCoarse, sandVeryFine,
sandFine, sandMedium, sandCoarse, sandVeryCoarse, gravelVeryFine, gravelFine,
gravelMedium, gravelCoarse, gravelVeryCoarse, cobblesSmall) VALUES " +
                                "(" + clayTotal + ", " + siltVeryFine + ", " +
siltFine + ", " + siltMedium + ", " + siltCoarse + ", " + siltVeryCoarse + ",
" + sandVeryFine + ", " + sandFine + ", " + sandMedium + ", " + sandCoarse +
", " + sandVeryCoarse + ", " + gravelVeryFine + ", " + gravelFine + ", " +
gravelMedium + ", " + gravelCoarse + ", " + gravelVeryCoarse + ", " +
cobblesSmall + ")";
                //Console.WriteLine(query);
                int grainSizeDistributionID = this.executeDBUpdate(query);

                return grainSizeDistributionID;
            }

            private void parseCrossSections(int dataID, Element crossSection,
int index)
            {

                String name = crossSection.getChildTextTrim("name");

                if(name.Equals("7977.576 (EC)")){
                    Console.WriteLine(name);
                }
                Single? station = this.parseSingle(crossSection
                        .getChildTextTrim("station"));
                Int32? leftBankTop = this.parseInt(crossSection
                        .getChildTextTrim("leftBankTop"));
                Int32? leftBankToe = this.parseInt(crossSection
                        .getChildTextTrim("leftBankToe"));
                Int32? rightBankTop = this.parseInt(crossSection
                        .getChildTextTrim("rightBankTop"));
                Int32? rightBankToe = this.parseInt(crossSection
                        .getChildTextTrim("rightBankToe"));
                Single? nBed =
this.parseSingle(crossSection.getChildTextTrim("nBed"));
                Single? nLeftBank = this.parseSingle(crossSection
                        .getChildTextTrim("nLeftBank"));
                Single? nRightBank = this.parseSingle(crossSection
                        .getChildTextTrim("nRightBank"));
                Single? nLeftFloodplain = this.parseSingle(crossSection
                        .getChildTextTrim("nLeftFloodplain"));
                Single? nRightFloodplain = this.parseSingle(crossSection
                        .getChildTextTrim("nRightFloodplain"));
                Single? bedrockElevation = this.parseSingle(crossSection
                        .getChildTextTrim("bedrockElevation"));
                Single? leftGroundWaterTable = this.parseSingle(crossSection
                        .getChildTextTrim("leftGroundwaterTable"));
```

```
                Single? rightGroundWaterTable = this.parseSingle(crossSection
                        .getChildTextTrim("rightGroundwaterTable"));

                String bedSediment = crossSection
                        .getChildTextTrim("bedSediment");
                String leftBankSoil = crossSection
                        .getChildTextTrim("leftBankSoil");
                String rightBankSoil = crossSection
                        .getChildTextTrim("rightBankSoil");


                Int32 bedSedimentID = xsidMap[bedSediment];
                Int32 leftBankSoilID = xsidMap[leftBankSoil];
                Int32 rightBankSoilID = xsidMap[rightBankSoil];

                String query = "INSERT INTO crosssection (name, dataID,
station, leftBankTop, leftBankToe, rightBankTop, rightBankToe, nBed,
nLeftBank, nRightBank, nLeftFloodplain, nRightFloodplain, bedRockElevation,
leftGroundwaterTable, rightGroundwaterTable, bedSedimentID, leftBankSoilID,
rightBankSoilID) "
                        + "VALUES ('"
                        + name
                        + "', "
                        + dataID
                        + ", "
                        + station
                        + ", "
                        + leftBankTop
                        + ", "
                        + leftBankToe
                        + ", "
                        + rightBankTop
                        + ", "
                        + rightBankToe
                        + ", "
                        + nBed
                        + ", "
                        + nLeftBank
                        + ", "
                        + nRightBank
                        + ", "
                        + nLeftFloodplain
                        + ", "
                        + nRightFloodplain
                        + ", "
                        + bedrockElevation
                        + ", "
                        + leftGroundWaterTable
                        + ", "
                        + rightGroundWaterTable
                        + ", "
                        + bedSedimentID
                        + ", "
                        + leftBankSoilID + ", " + rightBankSoilID + ")";
```

```
                // execute query and retrieve the generated key
                int crossSectionID = this.executeDBUpdate(query);

                mapXSIDString(crossSection, index, crossSectionID);
                // Place crossSection inside of HashMap using the db primary
key as the
                // key and a reference string for the channelModel as the
value

                Element transect = crossSection.getChild("transect");
                if (transect != null)
                {
                    parseTransect(crossSectionID, transect);
                }

            }

            private void parseTransect(int crossSectionID, Element transect)
            {
                String id =
this.parseString(transect.getAttributeValue("id"));
                String query = "INSERT INTO transect (crossSectionID, id) " +
"VALUES (" + crossSectionID + ", " + id
                    + ")";

                int transectID = this.executeDBUpdate(query);

                List<Element> siteList = transect.getChildren("sites");

                foreach(Element site in siteList){
                    this.parseSite(null, transectID, site);
                }
            }

            private void parseReaches(int dataID, Element reach, int index)
            {

                String name =
this.parseString(reach.getChildTextTrim("name"));

                String query = "INSERT INTO reaches (dataID, name) " +
"VALUES ("
                        + dataID + ", " + name + ")";

                int reachesID = this.executeDBUpdate(query);

                mapXSIDString(reach, index, reachesID);

                /*
                 * reach culvertSection bridgeSection dropStructure
genericStructure
                 * tributary lateralInflow
                 */
```

```
            }

            private void parseStructuresData(int dataID, Element structure)
            {

                String query = "INSERT INTO structures_data (dataID) VALUES
("  + dataID
                        + ")";

                int structuresDataID = this.executeDBUpdate(query);

                Element culverts = structure.getChild("culverts");
                List<Element> culvertSectionList = culverts
                        .getChildren("culvertSection");
                for (int i = 0; i < culvertSectionList.Count; i++)
                {
                    this.parseCulvertSections(structuresDataID,
                            culvertSectionList[i], i);
                }

                Element bridgeSections =
structure.getChild("bridgeSections");
                List<Element> bridgeSectionList = bridgeSections
                        .getChildren("bridgeSection");
                for (int i = 0; i < bridgeSectionList.Count; i++)
                {
                    this.parseBridgeSections(structuresDataID,
                            bridgeSectionList[i], i);
                }

                Element dropStructures =
structure.getChild("dropStructures");
                List<Element> dropStructureList = dropStructures
                        .getChildren("dropStructure");
                for (int i = 0; i < dropStructureList.Count; i++)
                {
                    this.parseDropStructures(structuresDataID,
                            dropStructureList[i], i);
                }

                Element genericStructures =
structure.getChild("genericStructures");
                List<Element> genericStructureList = genericStructures
                        .getChildren("genericStructure");
                for (int i = 0; i < genericStructureList.Count; i++)
                {
                    this.parseGenericStructures(structuresDataID,
                            genericStructureList[i], i);
                }

            }
```

```
            private void parseTributaries(int dataID, Element tributary, int
index)
            {

                String name =
this.parseString(tributary.getChildTextTrim("name"));
                String dataFile = this.parseString(tributary
                        .getChildTextTrim("dataFile"));

                String crossSection = tributary
                        .getChildTextTrim("crossSection");

                Int32 crossSectionID = xsidMap[crossSection];

                String query = "INSERT INTO tributaries (name, dataFile,
dataID, crossSectionID) VALUES ("
                        + name
                        + ", "
                        + dataFile
                        + ", "
                        + dataID
                        + ", "
                        + crossSectionID + ")";

                int tributaryID = this.executeDBUpdate(query);

                mapXSIDString(tributary, index, tributaryID);
            }

            private void parseLateralInflows(int dataID, Element
lateralInflow,
                    int index)
            {
                String name =
this.parseString(lateralInflow.getChildTextTrim("name"));
                String dataFile = this.parseString(lateralInflow
                        .getChildTextTrim("dataFile"));

                String upCrossSection = lateralInflow
                        .getChildTextTrim("upCrossSection");

                Int32 upCrossSectionID = xsidMap[upCrossSection];

                String downCrossSection = lateralInflow
                        .getChildTextTrim("downCrossSection");

                Int32 downCrossSectionID = xsidMap[downCrossSection];

                String query = "INSERT INTO lateralinflow (name, dataFile,
upCrossSectionID, downCrossSectionID) VALUES ("
                        + name
                        + ", "
                        + dataFile
                        + ", "
```

```
                        + upCrossSectionID
                        + ", "
                        + downCrossSectionID + ")";

                int lateralInflowID = this.executeDBUpdate(query);

                mapXSIDString(lateralInflow, index, lateralInflowID);
            }

            private void parseCulvertSections(int structuresDataID,
                    Element culvertSection, int index)
            {

                int structureID = this.parseStructure(culvertSection);
                Int32? chartNumber =
this.parseInt(culvertSection.getChildTextTrim("chartNumber"));
                Int32? scaleNumber =
this.parseInt(culvertSection.getChildTextTrim("scaleNumber"));
                Int32? numberOfBarrels =
this.parseInt(culvertSection.getChildTextTrim("numberOfBarrels"));
                Single? span =
this.parseSingle(culvertSection.getChildTextTrim("span"));
                Single? rise =
this.parseSingle(culvertSection.getChildTextTrim("rise"));
                Single? entranceLossCoefficient =
this.parseSingle(culvertSection.getChildTextTrim("entranceLossCoefficient"));

                String query = "INSERT INTO culvertSections
(structures_dataID, structureID, chartNumber, scaleNumber, numberOfBarrels,
span, rise) VALUES (" +
                structuresDataID + ", " + structureID + ", " + chartNumber +
", " + scaleNumber + ", " + numberOfBarrels + ", " + span + ", " + rise +
")";

                int culvertSectionID = this.executeDBUpdate(query);
                this.mapXSIDString(culvertSection, index, culvertSectionID);

            }

            private void parseBridgeSections(int structuresDataID,
                    Element bridgeSection, int index)
            {

                int structureID = this.parseStructure(bridgeSection);


                Single? width =
this.parseSingle(bridgeSection.getChildTextTrim("width"));
                Single? sideslope =
this.parseSingle(bridgeSection.getChildTextTrim("sideslope"));
                Single? pierWidth =
this.parseSingle(bridgeSection.getChildTextTrim("pierWidth"));
                Single? pierShapeCoefficient =
this.parseSingle(bridgeSection.getChildTextTrim("pierShapeCoefficient"));
```

```
                Single? pierLossCoefficient =
this.parseSingle(bridgeSection.getChildTextTrim("pierLossCoefficient"));


                String query = "INSERT INTO bridgeSections
(structures_dataID, structureID, width, sideslope, pierWidth,
pierShapeCoefficient, pierLossCoefficient) VALUES (" +
                structuresDataID + ", " + structureID + ", " + width + ", " +
sideslope + ", " + pierWidth + ", " + pierShapeCoefficient + ", " +
pierLossCoefficient + ")";


                int bridgeSectionID = this.executeDBUpdate(query);
                this.mapXSIDString(bridgeSection, index, bridgeSectionID);

            }

            private void parseDropStructures(int structuresDataID,
                    Element dropStructure, int index)
            {

                int structureID = this.parseStructure(dropStructure);
                Single? width =
this.parseSingle(dropStructure.getChildTextTrim("width"));
                Single? sideslope =
this.parseSingle(dropStructure.getChildTextTrim("sideslope"));
                Single? energyLossCoefficient =
this.parseSingle(dropStructure.getChildTextTrim("energyLossCoefficient"));


                String query = "INSERT INTO dropStructures
(structures_dataID, structureID, width, sideslope, energyLossCoefficient)
VALUES (" +
                structuresDataID + ", " + structureID + ", " + width + ", " +
sideslope + ", " + energyLossCoefficient + ")";


                int dropStructureID = this.executeDBUpdate(query);
                this.mapXSIDString(dropStructure, index, dropStructureID);

            }

            private void parseGenericStructures(int structuresDataID,
                    Element genericStructure, int index)
            {

                int structureID = this.parseStructure(genericStructure);

                Single? width =
this.parseSingle(genericStructure.getChildTextTrim("width"));
                Single? sideslope =
this.parseSingle(genericStructure.getChildTextTrim("sideslope"));
                Single? energyLossCoefficient =
this.parseSingle(genericStructure.getChildTextTrim("energyLossCoefficient"));
```

```
                Element ratingCurve =
genericStructure.getChild("ratingCurve");
                int ratingCurveID = this.parseRatingCurve(ratingCurve);


                String query = "INSERT INTO genericStructures
(structures_dataID, structureID, width, sideslope, energyLossCoefficient,
ratingCurveID) VALUES (" +
                structuresDataID + ", " + structureID + ", " + width + ", " +
sideslope + ", " + energyLossCoefficient + ", " + ratingCurveID + ")";


                int genericStructureID = this.executeDBUpdate(query);
                this.mapXSIDString(genericStructure, index,
genericStructureID);
            }

            private int parseRatingCurve(Element ratingCurve)
            {

                String query = "INSERT INTO ratingCurves() values()";

                int ratingCurveID = this.executeDBUpdate(query);

                //TODO:  ratingCurveSegments

                return ratingCurveID;
            }


            //this method actually returns the key generated because other
elements extend from this element
            private int parseStructure(Element structureType)
            {
                String name =
this.parseString(structureType.getChildTextTrim("name"));
                Single? station = this.parseSingle(structureType
                        .getChildTextTrim("dataFile"));
                Single? manningN = this.parseSingle(structureType
                        .getChildTextTrim("manningN"));
                Single? length = this

.parseSingle(structureType.getChildTextTrim("length"));
                Single? upstreamInvert = this.parseSingle(structureType
                        .getChildTextTrim("upstreamInvert"));
                Single? downstreamInvert = this.parseSingle(structureType
                        .getChildTextTrim("downstreamInvert"));
                Single? upstreamDrop = this.parseSingle(structureType
                        .getChildTextTrim("upstreamDrop"));
                Single? downstreamDrop = this.parseSingle(structureType
                        .getChildTextTrim("downstreamDrop"));
```

```
                String query = "INSERT INTO structure (name, station,
manningN, length, upstreamInvert, downstreamInvert, upstreamDrop,
downstreamDrop) VALUES ("
                        + name
                        + ", "
                        + station
                        + ", "
                        + manningN
                        + ", "
                        + length
                        + ", "
                        + upstreamInvert
                        + ", "
                        + downstreamInvert
                        + ", "
                        + upstreamDrop
                        + ", "
                        + downstreamDrop
                        + ")";

                int structureID = this.executeDBUpdate(query);

                return structureID;

            }


            private void parseRunControlData(int runDataID, Element
runControlData)
            {
                // TODO Auto-generated method stub

                String name =
this.parseString(runControlData.getChildTextTrim("name"));

                Boolean? doSedimentTransport =
this.parseBoolean(runControlData.getChildTextTrim("doSedimentTransport"));
                Boolean? doToeErosion =
this.parseBoolean(runControlData.getChildTextTrim("doToeErosion"));
                Boolean? doBankStability =
this.parseBoolean(runControlData.getChildTextTrim("doBankStability"));

                String simulationStartTime =
this.parseString(runControlData.getChildTextTrim("simulationStartTime"));
                String simulationEndTime =
this.parseString(runControlData.getChildTextTrim("simulationEndTime"));

                Int32? initialTimeStep =
this.parseInt(runControlData.getChildTextTrim("initialTimeStep"));
                String upstreamBC =
this.parseString(runControlData.getChildTextTrim("upstreamBC"));
                String downstreamBoundaryCondition =
this.parseString(runControlData.getChildTextTrim("downstreamBoundaryCondition
"));
```

```
                String downstreamWaterLevelFile =
this.parseString(runControlData.getChildTextTrim("downstreamWaterLevelFile"))
;
                String washLoadSizeClass =
this.parseString(runControlData.getChildTextTrim("washLoadSizeClass"));
                Single? cohesiveSiltClayFraction =
this.parseSingle(runControlData.getChildTextTrim("cohesiveSiltClayFraction"))
;
                Single? upstreamSedimentWeightCoefficient =
this.parseSingle(runControlData.getChildTextTrim("upstreamSedimentWeightCoeff
icient"));
                String upstreamSedimentBoundaryCondition =
this.parseString(runControlData.getChildTextTrim("upstreamSedimentBoundaryCon
dition"));

                Single? downstreamGradeControl =
this.parseSingle(runControlData.getChildTextTrim("downstreamGradeControl"));
                Int32? processesAnalyzedByBankStability =
this.parseInt(runControlData.getChildTextTrim("processesAnalyzedByBankStabili
ty"));
                Int32? numberOfShearEmergences =
this.parseInt(runControlData.getChildTextTrim("numberOfShearEmergences"));
                Single? tensionCrackDepth =
this.parseSingle(runControlData.getChildTextTrim("tensionCrackDepth"));
                Int32? numberOfSkippedTimeSteps =
this.parseInt(runControlData.getChildTextTrim("numberOfSkippedTimeSteps"));
                Single? blockRetentionTime =
this.parseSingle(runControlData.getChildTextTrim("blockRetentionTime"));
                Boolean? doGroundWaterDynamics =
this.parseBoolean(runControlData.getChildTextTrim("doGroundWaterDynamics"));

                String query = "INSERT INTO runControlData (runDataID, name,
doSedimentTransport, doToeErosion, doBankStability, simulationStartTime,
simulationEndTime, initialTimeStep, upstreamBC, downstreamBoundaryCondition,
downstreamWaterLevelFile, washLoadSizeClass, cohesiveSiltClayFraction,
upstreamSedimentWeightCoeff, upstreamSedimentBoundaryCond,
downstreamGradeControl, processesAnalyzedByBankStability,
numberOfShearEmergences, tensionCrackDepth, numberOfSkippedTimeSteps,
blockRetentionTime, doGroundWaterDynamics) VALUES (" + runDataID + ", " +
name + ", " + doSedimentTransport + ", " + doToeErosion + ", " +
doBankStability + ", " + simulationStartTime + ", " + simulationEndTime + ",
" + initialTimeStep + ", " + upstreamBC + ", " + downstreamBoundaryCondition
+ ", " + downstreamWaterLevelFile + ", " + washLoadSizeClass + ", " +
cohesiveSiltClayFraction + ", " + upstreamSedimentWeightCoefficient + ", " +
upstreamSedimentBoundaryCondition + ", " + downstreamGradeControl + ", " +
processesAnalyzedByBankStability + ", " + numberOfShearEmergences + ", " +
tensionCrackDepth + ", " + numberOfSkippedTimeSteps + ", " +
blockRetentionTime + ", " + doGroundWaterDynamics + ")";
                //Console.WriteLine(query);
                int runControlDataID = this.executeDBUpdate(query);

                /* TODO
                 * ratingCurve
                 * sedimentFraction
```

```
             *
             *
             *
             */


        }

        private void parseOutputOptions(int runDataID, Element
outputOptions)
        {
            // TODO Auto-generated method stub

        }

        private void parseScenarios(int runDataID, Element scenario)
        {
            // TODO Auto-generated method stub


        }

        private int executeDBUpdate(String query)
        {

            //Only insert if the user requests this
            if (!performInserts)
            {
                return -1;
            }

            //Console.WriteLine("Query before replace -  " + query);
            //REPLACE EMPTY STRINGS WITH NULL keyword
            //This means that we have to be very careful with spacing in
database queries
            query = query.Replace("(,", "(NULL,");
            query = query.Replace(" ,", " NULL,");
            query = query.Replace(" )", " NULL)");
            //Console.WriteLine("Executing -  " + query);
            int autoIncKey = -1;
            try
            {
                MySqlCommand sqlComm = new MySqlCommand(query + "; select
last_insert_id();", conn);
                autoIncKey = Convert.ToInt32(sqlComm.ExecuteScalar());

                return autoIncKey;
            }

            catch (Exception e)
            {
                Console.WriteLine(query + "\n" + e.Message);
                throw new Exception(query + "\n" + e.Message);
            }
```

```
            }

            // Construct the reference string for a given element to map and
maintain
            // many to many foreign key relationships
            // Parameter element is element reference string is being
constructed for
            // Index is location of element within parent tag
            private void mapXSIDString(Element element, int index, int
dbKeyValue)
            {

                // Get reference string to maintain proper map of foreign
keys
                String referenceKey = "." + index;
                while (element != null)
                {
                    referenceKey = "/@" + element.getName() + referenceKey;
                    element = element.getParentElement();
                }

                referenceKey = "#/" + referenceKey;

                //Console.WriteLine(referenceKey);

                xsidMap.Add(referenceKey, dbKeyValue);
            }

            private Int32? parseInt(String value)
            {
                if (value == null)
                    return null;
                else
                    return Int32.Parse(value);
            }

            private Single? parseSingle(String value)
            {
                if (value == null)
                    return null;
                else
                    return Single.Parse(value);
            }

            private String parseString(String value)
            {
                if (value == null)
                    return null;
                else
                    return "'" + value + "'";
            }

            private Boolean? parseBoolean(String value)
```

```
            {
                if (value == null)
                    return null;
                else
                    return Boolean.Parse(value);
            }

            private class XMLLoader
            {

                private XmlDocument xmlDoc;

                public XMLLoader()
                {
                    xmlDoc = new XmlDocument();
                }

                public Document build(String xmlFileName)
                {
                    //FileStream input = new FileStream(xmlFileName,
FileMode.Open);
                    xmlDoc.Load(xmlFileName);
                    return
                        new Document(xmlDoc);
                }

            }


            private class Document
            {
                private XmlDocument xmlDoc;

                public Document(XmlDocument xmlDoc)
                {
                    this.xmlDoc = xmlDoc;
                }

                public Element getRootElement()
                {
                    return new Element(xmlDoc.DocumentElement);
                }
            }

            private class Element
            {
                private XmlElement xmlElement;

                public Element(XmlElement xmlElement)
                {
                    this.xmlElement = xmlElement;
                }

                public String getName()
```

```
                {
                    return xmlElement.LocalName;
                }

                public Element getParentElement()
                {
                    if (xmlElement.ParentNode == null ||
xmlElement.ParentNode.GetType().ToString().Equals("System.Xml.XmlDocument"))
                        return null;
                    else
                        return new
Element((XmlElement)xmlElement.ParentNode);
                }

                public List<Element> getChildren(String name)
                {
                    List<Element> childList = new List<Element>();

                    XmlNodeList childNodeList =
xmlElement.GetElementsByTagName(name);

                    IEnumerator iterator = childNodeList.GetEnumerator();
                    while (iterator.MoveNext())
                    {
                        childList.Add(new
Element((XmlElement)iterator.Current));
                    }

                    return childList;

                }

                public String getTextTrim()
                {
                    return xmlElement.InnerText.Trim();
                }

                public String getText()
                {
                    return xmlElement.InnerText;
                }


                public String getChildText(String name)
                {
                    IEnumerator elements =
xmlElement.GetElementsByTagName(name).GetEnumerator();
                    elements.MoveNext();

                    if (elements.Current == null)
                        return null;
                    else
                        return ((XmlElement)elements.Current).InnerText;
```

```
                    //return
((XmlElement)(xmlElement.GetElementsByTagName(name).GetEnumerator().Current))
.Value;
                }


                public String getChildTextTrim(String name)
                {
                    IEnumerator elements =
xmlElement.GetElementsByTagName(name).GetEnumerator();
                    elements.MoveNext();

                    if (elements.Current == null)
                        return null;
                    else
                        return
((XmlElement)elements.Current).InnerText.Trim();
                    //return
((XmlElement)(xmlElement.GetElementsByTagName(name).GetEnumerator().Current))
.Value.Trim();
                }

                public Element getChild(String name)
                {
                    IEnumerator elements =
xmlElement.GetElementsByTagName(name).GetEnumerator();
                    elements.MoveNext();

                    if (elements.Current == null)
                        return null;
                    else
                        return new Element((XmlElement)elements.Current);
                    //return new
Element((XmlElement)xmlElement.GetElementsByTagName(name).GetEnumerator().Cur
rent);
                }

                public String getAttributeValue(String name)
                {
                    return xmlElement.GetAttribute(name);
                }

            }

        }
}
```

DEMFILEPARSER

Written by Chenchutta Denaye Cross Jackson
Version 1
Copy write July 2012
The DEM file parser extracts data from the DEM file. The DEM file format is based on the ERSI file format and contains six fields in its header.
The DEM file contains z (elevation) values. From the DEM file the x (easting) and y (northing) values are determined by the z value's position in the DEM file.
The idea is to use this information from the raw HeightMap to create a virtual reality terrain.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
using System.IO;
using System.Drawing;
using System.Windows.Forms;
using System.Drawing.Imaging;

namespace Concepts3D
{
    class GridFileParser
    {

        private int ncols = 0;
        private int nrows = 0;
        private int cellSize = 0;
        private Single xllcorner = 0.0f;
        private Single yllcorner = 0.0f;
        private Single NODATAVALUE = 0.0f;

        private Single highestElevation = -1;
        private Single lowestElevation = -1;
        private int datacols = 0;
        private int datarows = 0;
        private Single[,] elevationData = null;
        private List<Point3D> pointList = new List<Point3D>();

        private Hashtable yAxisMap = new Hashtable();
        private Hashtable xAxisMap = new Hashtable();

        private Single elevationStartRange = -1;

        public GridFileParser(String inputFile)
        {
            parse(inputFile);
```

```
        }

        public void parse(String inputFile)
        {

            // TODO Auto-generated method stub
               String[] line;

                  if (inputFile == null) {
                throw new Exception("Input file is null");
                  }

            StreamReader reader = new StreamReader(inputFile);
                  if (reader.Peek() > -1) {
                        line = reader.ReadLine().Split(new char[] {' '},
StringSplitOptions.RemoveEmptyEntries);//.Split(" +");
                        ncols = Int32.Parse(line[1]);
                  }
            if (reader.Peek() > -1)
            {
                        line = reader.ReadLine().Split(new char[] {' '},
StringSplitOptions.RemoveEmptyEntries);
                        nrows = Int32.Parse(line[1]);
                  }
            if (reader.Peek() > -1)
            {
                        line = reader.ReadLine().Split(new char[] {' '},
StringSplitOptions.RemoveEmptyEntries);
                        xllcorner = Single.Parse(line[1]);
                  }
            if (reader.Peek() > -1)
            {
                        line = reader.ReadLine().Split(new char[] {' '},
StringSplitOptions.RemoveEmptyEntries);
                        yllcorner = Single.Parse(line[1]);
                  }
            if (reader.Peek() > -1)
            {
                        line = reader.ReadLine().Split(new char[] {' '},
StringSplitOptions.RemoveEmptyEntries);
                        cellSize = Int32.Parse(line[1]);
                  }
            if (reader.Peek() > -1)
            {
                        line = reader.ReadLine().Split(new char[] {' '},
StringSplitOptions.RemoveEmptyEntries);
                        NODATAVALUE = Single.Parse(line[1]);
                  }

            elevationData = new Single[nrows, ncols];

            Console.WriteLine("ncols: " + ncols + "\nnrows: " + nrows
                    + "\nxllcorner: " + xllcorner + "\nyllcorner: " +
yllcorner
```

```
                    + "\ncellsize: " + cellSize);

                Single x = xllcorner - cellSize;
                Single y = yllcorner + nrows - cellSize;
                Single z = 0.0f;

                int count = 0;

            //Single[] rowMap = new Single[nrows];  //keeps track of row
order so it can easily be reversed
            //int rowMapIndex = 0;
            Single topMostRow = -1.0f;


            while (reader.Peek() > -1)
            {
                        line = reader.ReadLine().Split(new char[] {' '},
StringSplitOptions.RemoveEmptyEntries);

                        for (int i = 0; i < line.Length; i++) {
                              x = x + cellSize;


                              if (x % (ncols + xllcorner) == 0) {
                                    x = xllcorner;
                                    y = y - cellSize;
                              }

                    if (count == 0){
                        topMostRow = y;
                    }

                              z = Single.Parse(line[i]);
                    elevationData[(int)(count / ncols), (int)(count % ncols)]
= z;

                    if (lowestElevation == -1 && z != -9999)
                        lowestElevation = z;
                    else if (z < lowestElevation && z != -9999)
                        lowestElevation = z;

                    if (z > highestElevation)
                        highestElevation = z;

                    //rowMap[rowMapIndex++] = y;
                              Point3D point = new Point3D(x, y, z);
                              pointList.Add(point);

                    if (!yAxisMap.ContainsKey((int)y))
                    {
                        yAxisMap.Add((int)y, (int)(count / ncols));
                    }

                    if (!xAxisMap.ContainsKey((int)x))
```

```
                    {
                        xAxisMap.Add((int)x, (int)(count % ncols));
                    }

                    //points[y,x] = z;
                    count++;
                              //Console.WriteLine(point.toString());

                        }
                }

            reader.Close();

            datarows = nrows;
            datacols = ncols;

            //PERFORM INTERPOLATION IF REQUESTED OR NEEDED
            if (cellSize >= 2)
            {
                InterpolationSelectionForm form = new
InterpolationSelectionForm(cellSize);
                form.ShowDialog();

                if (! form.interpolationTechnique.Equals("NONE"))
                {
                    datarows = (nrows - 1) * cellSize + 1;
                    datacols = (ncols - 1) * cellSize + 1;
                }

                if (form.interpolationTechnique.Equals("BILINEAR"))
                {
                    performBilinearInterpolation();
                }
                else if (form.interpolationTechnique.Equals("BARYCENTRIC"))
                {
                    performBarycentricInterpolation();
                }
            }

        }



        //Values must be read in starting at the last row of data values
because of different coordinate mapping
        //elevationStartRange indicates whether or not the range of elevation
values should be modified for better viewing
        //A value of 25 indicates that the range should start at 25, so if
values are actually 75-150 they would instead be
        //25-100
        //The lower the elevationStartRange, the higher water will appear
        public void createRaw32File(Rectangle selectedRegion, String
raw32FileName, Single elevationStartRange)
        {
```

```
            this.elevationStartRange = elevationStartRange;

            if (selectedRegion.Top < 0 || selectedRegion.Left < 0)
            {
                selectedRegion = new Rectangle(0, 0, selectedRegion.Right,
selectedRegion.Bottom);
            }

            try
            {

                FileInfo file = new FileInfo(raw32FileName);
                FileStream s = file.Open(FileMode.Create, FileAccess.Write);

                BinaryWriter bs = new BinaryWriter(s);

                Single heightValue = -1.0f;


                Bitmap testImage2 = new Bitmap(selectedRegion.Width,
selectedRegion.Height);
                int bitMapX2 = -1;
                int bitMapY2 = -1;
                for (int y = selectedRegion.Top; y < selectedRegion.Top +
selectedRegion.Height; y++)
                {
                    bitMapY2++;
                    bitMapX2 = -1;
                    for (int x = selectedRegion.Left; x < selectedRegion.Left
+ selectedRegion.Width; x++)
                    {
                        //Console.WriteLine("[" + x + ", " + y + "]");
                        heightValue = elevationData[y, x];
                        if (heightValue > 7000)
                        {
                            Console.WriteLine(heightValue);
                        }

                        if (elevationStartRange != -1)
                            heightValue = heightValue - lowestElevation +
elevationStartRange;

                        bs.Write(heightValue);

                        bitMapX2++;

                        if (heightValue < 0)
                        {
                            heightValue = 0;
                        }

                        testImage2.SetPixel(bitMapX2, bitMapY2,
Color.FromArgb((int)(heightValue), (int)(heightValue), (int)(heightValue)));
```

```
                    }
                }

                //testImage2.Save(raw32FileName);

                bs.Close();
                s.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
        }

        public Bitmap getBitmap()
        {
            Single imageRange = (highestElevation - lowestElevation);//
*METERSCALEUP;
            Console.WriteLine("Highest elevation is " + highestElevation);
            Console.WriteLine("Lowest elevation is " + lowestElevation);
            Console.WriteLine("Image range is " + imageRange);

            Single heightMapRange = 255;

            Single heightMapRatio = heightMapRange / imageRange;

            Bitmap image1 = null;

            //try
            //{

            image1 = new Bitmap(datacols, datarows);
            int elevation = -1;


            Single value = -1.0f;

            for (int y = 0; y < datarows; y++)
            {

                for (int x = 0; x < datacols; x++)
                {
                    value = elevationData[y, x];
                    if (value >= lowestElevation && value <=
highestElevation) //We attempt to weed out the NO DATA VALUES and possible
interpolated values influenced by the NO DATA VALUES
                    {
                        elevation = (Int32)((value - lowestElevation) *
heightMapRatio);

                        //int imageX = (Int32)(point.getX() - xllcorner);
                        //int imageY = (Int32)(point.getY() - yllcorner);
                        Color newColor = Color.FromArgb(elevation, elevation,
elevation);
```

```
                        //if (elevation > 15)
                        //{
                        //    Console.WriteLine(elevation);
                        //}
                        image1.SetPixel(x, y, newColor);
                    }
                }


            }

            return image1;
        }


        public void mergeSecondaryData(List<Point3D> secondaryPointList)
        {
            Single elevationValue = -1;
            int zeroCount = 0;
            foreach (Point3D point in secondaryPointList)
            {
                elevationValue = point.getZ();// -lowestElevation +
elevationStartRange;

                if (elevationValue != 0)
                {
                    if (elevationValue < lowestElevation)
                    {
                        lowestElevation = elevationValue;
                    }
                    if (elevationValue > highestElevation)
                    {
                        highestElevation = elevationValue;
                    }
                }
                else
                {
                    zeroCount++;
                    Console.WriteLine("(" + (int)point.getX() + ", " +
(int)point.getY() + ")");
                }

                int y = (int)yAxisMap[(int)point.getY()];
                int x = (int)xAxisMap[(int)point.getX()];
                elevationData[y, x] = elevationValue;
            }

            Console.WriteLine("There was a total of " + zeroCount + " zero
values found");
        }
```

```
        private void performBilinearInterpolation()
        {

            Single[,] interpolatedData = new Single[datarows, datacols];

            //two outer loops are for original data

            Single x0 = -1, y0 = -1, y1 = -1, x1 = -1;
            Single height00 = -1, height01 = -1, height10 = -1, height11 = -
1;

            for (int datay = 0; datay < nrows - 1; datay++)
            {
                //y1 = y0 + 1;
                for (int datax = 0; datax < ncols - 1; datax++)
                {

                    height00 = elevationData[datay, datax];
                    height01 = elevationData[datay + 1, datax];
                    height10 = elevationData[datay, datax + 1];
                    height11 = elevationData[datay + 1, datax + 1];


                    //two inner loops are for new created data
                    //an extra column and row of data will be interpolated
even though it should already have its values
                    //this might need to be fixed
                    y0 = datay * cellSize;
                    x0 = datax * cellSize;
                    y1 = (datay + 1) * cellSize;
                    x1 = (datax + 1) * cellSize;
                    for (int y = (int)y0; y < y1; y++)
                    {
                        for (int x = (int)x0; x < x1; x++)
                        {

                            Single sum1 = ( ((x1 - x) * (y1 - y)) / ((x1 -
x0) * (y1 - y0)) ) * height00;
                            Single sum2 = ( ((x - x0) * (y1 - y)) / ((x1 -
x0) * (y1 - y0)) ) * height10;
                            Single sum3 = ( ((x1 - x) * (y - y0)) / ((x1 -
x0) * (y1 - y0)) ) * height01;
                            Single sum4 = ( ((x - x0) * (y - y0)) / ((x1 -
x0) * (y1 - y0)) ) * height11;

                            Single value = sum1 + sum2 + sum3 + sum4;

                            interpolatedData[y, x] = value;

                        }

                    }

                }
```

```
            }

            elevationData = interpolatedData;

        }

        private void performBarycentricInterpolation()
        {

            Single[,] interpolatedData = new Single[datarows, datacols];

            //two outer loops are for original data

            Single x0 = -1, y0 = -1, y1 = -1, x1 = -1;
            //Single height00 = -1, height01 = -1, height10 = -1, height11 =
-1;

            for (int datay = 0; datay < nrows - 1; datay++)
            {
                //y1 = y0 + 1;
                for (int datax = 0; datax < ncols - 1; datax++)
                {
                    /*
                            * a--------------c
                            * |  \            |
                            * |     \         |
                            * |        \ V    |
                            * |           \   |
                            * b--------------d
                            */
                    Single heighta = -1, heightb = -1, heightc = -1, heightd
= -1;
                    heighta = elevationData[datay, datax];
                    heightb = elevationData[datay + 1, datax];
                    heightc = elevationData[datay, datax + 1];
                    heightd = elevationData[datay + 1, datax + 1];


                    //two inner loops are for new created data
                    //an extra column and row of data will be interpolated
even though it should already have its values
                    //this might need to be fixed
                    y0 = datay * cellSize;
                    x0 = datax * cellSize;
                    y1 = (datay + 1) * cellSize;
                    x1 = (datax + 1) * cellSize;

                    //This should only need to be calculated once since the
points for a regular grid
                    Single entireTriangleArea = Math.Abs((x0 * (y1 - y1) + x0
* (y1 - y0) + x1 * (y0 - y1)) / 2);

                    int xincrease = 1;
                    for (int y = (int)y0; y < y1; y++)
```

```
                    {
                        for (int x = (int)x0; x < x0 + xincrease; x++)
                        {
                            /*
                             * a--------------c
                             * |  \            |
                             * |     \         |
                             * |        \ V    |
                             * |           \   |
                             * b--------------d
                             */

                            //Get areas of each individual triangle opposite
the corresponding vertex (i.e. Ab = area of triangle opposite vertex b)
                            Single Aa = Math.Abs((x * (y1 - y1) + x0 * (y1 -
y) + x1 * (y - y1)) / 2);
                            Single Ab = Math.Abs((x0 * (y - y1) + x * (y1 -
y0) + x1 * (y0 - y)) / 2);
                            Single Ad = Math.Abs((x0 * (y1 - y) + x0 * (y -
y0) + x * (y0 - y1)) / 2);

                            Single value = ( (Aa * heighta) + (Ab * heightb)
+ (Ad * heightd) ) / entireTriangleArea;
                            interpolatedData[y, x] = value;
                        }

                        xincrease++;
                    }

                    int xstart = (int)x0;
                    for (int y = (int)y0; y < y1; y++)
                    {
                        for (int x = xstart; x < x1; x++)
                        {
                            /*
                             * a--------------c
                             * |  \            |
                             * |     \         |
                             * |        \ V    |
                             * |           \   |
                             * b--------------d
                             */

                            //Get areas of each individual triangle opposite
the corresponding vertex (i.e. Ab = area of triangle opposite vertex b)
                            Single Aa = Math.Abs((x * (y0 - y1) + x1 * (y1 -
y) + x1 * (y - y0)) / 2);
                            Single Ac = Math.Abs((x0 * (y - y1) + x * (y1 -
y0) + x1 * (y0 - y)) / 2);
                            Single Ad = Math.Abs((x0 * (y - y0) + x * (y0 -
y0) + x1 * (y0 - y)) / 2);

                            Single value = ((Aa * heighta) + (Ac * heightc) +
(Ad * heightd)) / entireTriangleArea;
```

```
                            interpolatedData[y, x] = value;

                            //Console.WriteLine("(" + x + ", " + y + ")");
                        }

                        xstart++;
                    }


                }
            }

            elevationData = interpolatedData;

        }


        private void performTriangulatedInterpolation()
        {

            Single[,] interpolatedData = new Single[datarows, datacols];

            //two outer loops are for original data

            Single x0 = -1, y0 = -1, x1 = -1, y1 = -1, x2 = -1, y2 = -1;
            Single heightA = -1, heightB = -1, heightC = -1;//, height11 = -
1;

            for (int datay = 0; datay < nrows - 1; datay++)
            {
                //y1 = y0 + 1;
                for (int datax = 0; datax < ncols - 1; datax++)
                {

                    Single determinant;
                    //two inner loops are for new created data
                    //an extra column and row of data will be interpolated
even though it should already have its values
                    //this might need to be fixed
                    y0 = datay * cellSize;
                    x0 = datax * cellSize;
                    y1 = (datay + 1) * cellSize;
                    x1 = (datax + 1) * cellSize;
                    y2 = (datay + 1) * cellSize;
                    x2 = datax * cellSize;

                    determinant = (x0 * y1) - (x1 * y0) + (x1 * y2) - (x2 *
y1) + (x2 * y0) - (x0 * y2);

                    heightA = elevationData[datay, datax];
                    heightB = elevationData[datay + 1, datax + 1];
                    heightC = elevationData[datay + 1, datax];
                    //height11 = elevationData[datay + 1, datax + 1];
```

```
                    int xFactor = 0;
                    for (int y = (int)y0; y < y1; y++)
                    {
                        xFactor++;
                        for (int x = (int)x0; x < x0 + xFactor; x++)
                        {

                            Single A      = ((y1-y2)*heightA+(y2-
y0)*heightB+(y0-y1)*heightC) / determinant;
                            Single B      = ((x2-x1)*heightA+(x0-
x2)*heightB+(x1-x0)*heightC) / determinant;
                            Single C      = ((x1*y2-x2*y1)*heightA+(x2*y0-
x0*y2)*heightB+(x0*y1-x1*y0)*heightC) / determinant;
                            Single value = (A * x) + (B * y) + C;

                            interpolatedData[y, x] = value;


                        }

                    }


                    y2 = datay * cellSize;
                    x2 = (datax + 1) * cellSize;

                    heightA = elevationData[datay, datax];
                    heightB = elevationData[datay + 1, datax + 1];
                    heightC = elevationData[datay, datax + 1];
                    //height11 = elevationData[datay + 1, datax + 1];

                    xFactor = cellSize;
                    for (int y = (int)y0; y < y1; y++)
                    {
                        xFactor--;
                        for (int x = (int)x0 + xFactor; x > 0; x--)
                        {

                         //Single A = ((y1 - y2) * heightA + (y2 - y0) *
heightB + (y0 - y1) * heightC) / determinant;
                            //Single B = ((x2 - x1) * heightA + (x0 - x2) *
heightB + (x1 - x0) * heightC) / determinant;
                            //Single C = ((x1 * y2 - x2 * y1) * heightA + (x2
* y0 - x0 * y2) * heightB + (x0 * y1 - x1 * y0) * heightC) / determinant;
                            //Single value = (A * x) + (B * y) + C;
                            Single A = ((y1 - y2) * heightA + (y2 - y0) *
heightB + (y0 - y1) * heightC) / determinant;
                            Single B = ((x2 - x1) * heightA + (x0 - x2) *
heightB + (x1 - x0) * heightC) / determinant;
                            Single C = ((x1 * y2 - x2 * y1) * heightA + (x2 *
y0 - x0 * y2) * heightB + (x0 * y1 - x1 * y0) * heightC) / determinant;
                            Single value = (A * x) + (B * y) + C;

                            interpolatedData[y, x] = value;
```

```
                    }

                }

            }
        }

        elevationData = interpolatedData;

    }


    public List<Point3D> get3DPointList()
    {
        return this.pointList;
    }


}
}
```

X-SECTIONPARSER

Written by: Chenchutta Cross Jackson
Copy write July 2012
Version 1
The X-Section parser component starts by reading the file and extracting each value in the line of the comma separated file. Once this data is extracted, it is stored in a list of a user defined Point3D objects. The Point3D class has x (easting), y (northing), and z (elevation properties for storing data). This data is merged with the DEM file data by looping through the POINT3D list merging the PointList with the DEM information.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace Concepts3D
{
    class CrossSectionParser
    {

        private List<Point3D> pointList = new List<Point3D>();

        public CrossSectionParser(String inputFile)
        {
            parse(inputFile);
        }

        public void parse(String inputFile)
        {

            String[] line;

            if (inputFile == null)
            {
                throw new Exception("Input file is null");
            }


            StreamReader reader = new StreamReader(inputFile);
            int lineCount = -1;
            int elevationPosition = -1;  //corresponds to the column where
elevation values are
            int eastingPosition = -1;  //corresponds to the column where
easting values are
            int northingPosition = -1;  //corresponds to the column where
northing values are

            while (reader.Peek() > -1)
            {
```

```
                line = reader.ReadLine().Split(new char[]{','},
StringSplitOptions.RemoveEmptyEntries);

                if (line[0].ToUpper().Equals("END"))
                {
                    break;
                }

                lineCount++;

                //READ HEADER AND FIND COLUMN NUMBER FOR EACH VALUE
                if (lineCount == 0)
                {
                    for (int i = 0; i < line.Length; i++)// (String s in
line)
                    {
                        String s = line[i];
                        if (s.ToUpper().Equals("ELEV"))
                        {
                            elevationPosition = i;
                        }
                        else if (s.ToUpper().Equals("EASTING"))
                        {
                            eastingPosition = i;
                        }
                        else if (s.ToUpper().Equals("NORTHING"))
                        {
                            northingPosition = i;
                        }
                    }

                }//end if
                else
                {
                    Single x = Single.Parse(line[eastingPosition]); //easting
                    Single y = Single.Parse(line[northingPosition]);
//northing
                    Single z = Single.Parse(line[elevationPosition]);//
*0.3048f;  //elevation

                    //add to 3D point list
                    Point3D point = new Point3D(x, y, z);
                    pointList.Add(point);
                    //Console.WriteLine(lineCount + ". " + point.toString());

                }
            }//end while

        }//end parse method

        public List<Point3D> getPointList()
        {
            return pointList;
        }
```

```
    }
}
```

CONCEPTS3D

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;
using System.Diagnostics;
using System.Threading;



namespace Concepts3D
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();

        }

        private const String OPENSIMHOME = @"\Concepts Project\opensim-
0.7.3.1\bin";

        //private IntPtr viewerHandle, openSimHandle;
        private Process viewerProcess = null;
        private Process openSimProcess = null;

        private void btnGo_Click(object sender, EventArgs e)
        {
            String primaryInputFile = txtSelectedFile.Text;
            String secondaryInputFile = txtSelectedCrossSectionFile.Text;

            DateTime oldDate = DateTime.Now;

            //PrimaryParser primaryParser;

            //Select parser based on file extension
            if(primaryInputFile.EndsWith(".txt", true, null))
            {
                GridFileParser parser = new GridFileParser(primaryInputFile);
```

```
                CrossSectionParser secondaryParser = null;

                if (secondaryInputFile != null &&
!secondaryInputFile.Equals(""))
                {
                    secondaryParser = new
CrossSectionParser(secondaryInputFile);

parser.mergeSecondaryData(secondaryParser.getPointList());
                }


                //Create a bitmap based on loaded data
                Bitmap bmp = parser.getBitmap();
                //Save the bitmap to disk
                //String outName = "C:\\bmpout3.bmp";
                //bmp.Save(@outName, ImageFormat.Bmp);

                DateTime newDate = DateTime.Now;
                Console.WriteLine("Executed in " + (newDate -
oldDate).TotalMilliseconds + "ms");

                //Load the bitmap into the image viewer so user can select
the region for the raw data file
                ImageViewer viewer = new ImageViewer(bmp);
                viewer.ShowDialog();

                //Get user selected region from viewer
                Rectangle selectedRegion = viewer.getSelectedRegion();

                //Create raw file data, we start the range at a number based
on the water level being 20 meters
                try
                {
                    parser.createRaw32File(selectedRegion, OPENSIMHOME +
@"\OUTPUT.f32", 5.0f);

                    MessageBox.Show("RAW32 Heightmap File Created!");
                }
                catch (Exception ex)
                {
                    throw ex;
                }

            }
            else if (primaryInputFile.EndsWith(".xml", true, null))
            {
                ConceptsXMLParser parser = new
ConceptsXMLParser(primaryInputFile);

            }
            else
            {
                MessageBox.Show("The file extension is not supported.");
```

```
            }

        }

        private void enableCrossSectionSelection(Boolean enable){
            if (enable)
            {

                lblCrossSection.Enabled = true;
                txtSelectedCrossSectionFile.Enabled = true;
                btnCrossSectionBrowse.Enabled = true;
            }
            else
            {
                lblCrossSection.Enabled = false;
                txtSelectedCrossSectionFile.Text = "";
                txtSelectedCrossSectionFile.Enabled = false;
                btnCrossSectionBrowse.Enabled = false;
            }
        }

        private void btnBrowse_Click(object sender, EventArgs e)
        {
            openFileDialog1.Title = "Please select a DEM file or CONCEPTS XML
file";
            openFileDialog1.FileName = @"C:\Concepts
Project\SampleData\dem_1m.txt";
            openFileDialog1.Filter = " DEM File (*.txt)|*.txt|CONCEPTS XML
File (*.xml)|*.xml";
            DialogResult dResult = this.openFileDialog1.ShowDialog();
            if (dResult == DialogResult.OK)
            {
                txtSelectedFile.Text = openFileDialog1.FileName;
            }

            btnGo.Enabled = true;
        }


        private void txtSelectedFile_TextChanged(object sender, EventArgs e)
        {
            txtSelectedCrossSectionFile.Text = "";

            if (txtSelectedFile.Text.EndsWith(".txt"))
            {
                enableCrossSectionSelection(true);
            }
            else if (txtSelectedFile.Text.EndsWith(".xml"))
            {
                enableCrossSectionSelection(false);
            }
        }

        private void btnCrossSectionBrowse_Click(object sender, EventArgs e)
```

```
        {
            openFileDialog2.Title = "Please select a Cross Section File";
            openFileDialog2.FileName = @"\SampleData\chat-xs-points.txt";
            openFileDialog2.Filter = " Cross Section File (*.txt)|*.txt";
            DialogResult dResult = this.openFileDialog2.ShowDialog();
            if (dResult == DialogResult.OK)
            {
                txtSelectedCrossSectionFile.Text = openFileDialog2.FileName;
            }

            btnGo.Enabled = true;
        }

        private void btnStartOpenSim_Click(object sender, EventArgs e)
        {

            String program = "cmd.exe";
            String args = "/k" + " cd " + OPENSIMHOME + " && " +
"OpenSim.exe";

            openSimProcess = Process.Start(program, args);
            Thread.Sleep(100);

            SetParent(openSimProcess.MainWindowHandle,
panelGameEngine.Handle);
            MoveWindow(openSimProcess.MainWindowHandle, 0, 0,
panelGameEngine.Width, panelGameEngine.Height, true);
        }

        private void btnOpenViewer_Click(object sender, EventArgs e)
        {

            String program = @"C:\Program Files\Imprudence\imprudence.exe";
            String args = "--settings settings_imprudence.xml";

            viewerProcess = Process.Start(program, args);
            //Thread.Sleep(20000); // Allow the process to open it's window

            //IntPtr newParent;
            viewerProcess.WaitForInputIdle();

            SetParent(viewerProcess.MainWindowHandle, panelViewer.Handle);
            MoveWindow(viewerProcess.MainWindowHandle, 0, 0,
panelViewer.Width, panelViewer.Height, true);

            //viewerProcess.WaitForInputIdle();

        }



        //WINDOWS API FUNCTIONS
        [DllImport("user32.dll")]
```

```
        static extern IntPtr SetParent(IntPtr hWndChild, IntPtr
hWndNewParent);

        [DllImport("user32.dll", EntryPoint = "SetWindowPos")]
        public static extern IntPtr SetWindowPos(IntPtr hWnd, int
hWndInsertAfter, int x, int Y, int cx, int cy, int wFlags);

        [DllImport("user32.dll", SetLastError = true)]
        private static extern bool MoveWindow(IntPtr hwnd, int x, int y, int
cx, int cy, bool repaint);

        [DllImport("Kernel32.dll")]
        static extern Boolean AllocConsole();

        [DllImport("user32.dll", EntryPoint = "FindWindow")]
        public static extern IntPtr FindWin(string lpClassName, string
lpWindowName);

        [DllImport("user32.dll", EntryPoint = "GetParent")]
        public static extern IntPtr GetParent(IntPtr hWnd);

        [DllImport("user32.dll", CharSet = CharSet.Auto, ExactSpelling =
true)]
        public static extern bool IsChild(IntPtr hWndParent, IntPtr hwnd);

        //Clean up processes when the main form closes
        private void MainForm_FormClosing(object sender, FormClosingEventArgs
e)
        {
            if (viewerProcess != null)
            {
                //Make sure the process is stopped and its resources are
released
                viewerProcess.CloseMainWindow();
                viewerProcess.Kill();
                viewerProcess.Dispose();
            }
            if (openSimProcess != null)
            {
                //Make sure the process is stopped and its resources are
released
                openSimProcess.CloseMainWindow();
                openSimProcess.Kill();
                openSimProcess.Dispose();
            }
        }


    }
}
```

VITA

Chenchutta Cross Jackson was born and raised in Itta Bena, MS. She is the daughter of Mary Cross who is also of Itta Bena, MS. She attended Leflore County High School where she graduated in 1996. In the spring of 1997 she attended Mississippi Valley State University (MVSU) in Itta Bena, MS. While pursuing her undergraduate studies she was affiliated with many honor programs and groups such as the MVSU collegiate cheerleading squad, Alpha Kappa Mu, Beta Kappa Chi, National Honor Society, Who's Who among College Students, and the Mathematics and Computer Science Club. In May 2000 she graduated magna cum laude with a bachelor's degree in Computer and Information Sciences. In the summer of 2000 she had the opportunity to work as an intern at North Carolina State University under the direction of Dr. Dave McAllister through an Alliance for Graduate Education and the Professoriate (AGEP) program. At North Carolina State University she worked on one of the first virtual reality environments used to model flight simulations. In the summer of 2001 she was accepted into the graduate program at the University of Mississippi. During the summer of 2001 she accepted the opportunity to be a part of the Alliance for Graduate Education in Mississippi (AGEM) Summer Research Institute for Undergraduate (SRIU) program. Through this research arm she expanded her knowledge in the areas of computer graphics and robotics under the mentorship of Dr. Pamela Lawhead. While attending graduate school in the fall of 2002, she was also employed at the United States Department of Agriculture. She was employed for 12 years under the

supervision of Dr. Carlos Alonzo and Dr. Eddy Langendoen in the Watershed Physical Processes Research Unit as an Information Systems Specialist.